Bases de données: Introduction et Objectifs

I. Introduction

Les entreprises gèrent des volumes de données très grands

- Giga, Terra, Péta –octets
- Numériques, Textuelles, Multi-média (images, films,...)

Il faut pouvoir facilement

- Archiver les données sur mémoires secondaires permanente
- Retrouver les données pertinentes à un traitement
- Mettre à jour les données variant dans le temps

Les données sont structurées et identifiées

- Données élémentaires ex: Votre salaire, Votre note en BD
- Données composées ex: Votre CV, vos résultats de l'année
- Identifiant humain ex: NSS ou machine: P26215

Qu'est-ce qu'une BD ?

- Collection de données structurées reliées par des relations
- Interrogeable et modifiable par des langages de haut niveau

La hiérarchie des mémoires

Capacité
vs
Coût &
Vitesse

Mémoire
terciaire

5-10 ms

Mémoire
secondaire

80-200 ns

Mémoire principale
(RAM)

3-10 ns

Cache (SRAM)

2-5 ns

Registres

Un accès disque est environ 100,000 fois plus lent qu'un accès mémoire!



- Eviter les accès disques
  - grande mémoire principale
- Amortir les accès disques
  - placement des données
- Minimiser le nombre d'accès disques
  - méthodes d'accès

Un peu d'histoire

Années 60:

- Récipients logique de données ➔ fichiers sur dique
- Accès séquentiel puis sur clé
  - Lire (Nomf, Article), Ecrire (Nomf, Article)
  - Lire (Nomf, Article, Clé), Ecrire (Nomf, article, Clé)

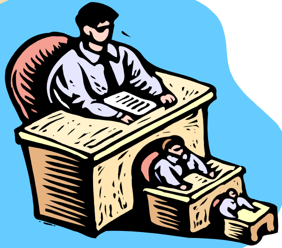
Années 70:

- Avènement des Bases de Données Réseaux (BD)
- Ensemble de fichiers reliés par des pointeurs
- Langage d'interrogation par navigation

Années 80:

- Avènement des Bases de Données Relationnelles (BDR)
- Relations entre ensemble de données
- Langage d'interrogation par assertion logique

Systèmes de fichiers



Comptabilité



Chirurgie

Consultations



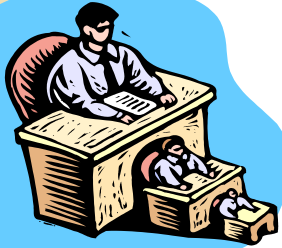
Psychiatrie



Caractéristiques

Problèmes

Format des fichiers



Dupont

Symptomes : y
Turlututu : sqj
Symptomes : y
Turlututu : sdd
Analyses : xxx

Dupond

Turlututusqjsk
Symptom: yyyy
Analyses xxxx

Turlututudhjsd
Analyses :xx



Duhpon

Symptomes : yy
Analyses : xxxx

Symptomes : yy

Duipont

Turlututu : sq

Symptomyyyy
Analysesxxxx

Turlututudhjsd



Caractéristiques

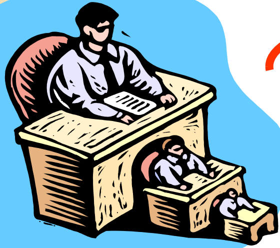
Plusieurs applications

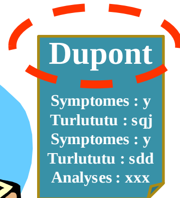
- plusieurs formats
- plusieurs langages

Problèmes

➔ **Difficultés de gestion**

Redondance (données)

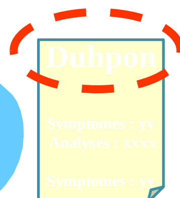


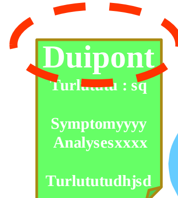














Caractéristiques

Plusieurs applications

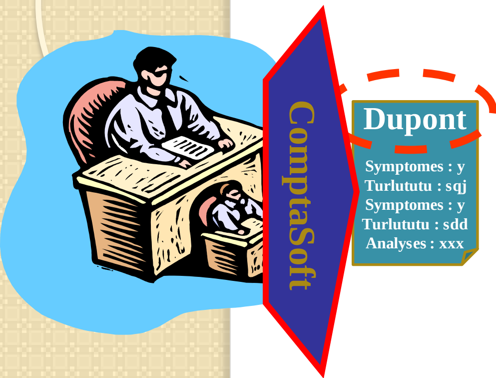
- ➔ plusieurs formats
- ➔ plusieurs langages

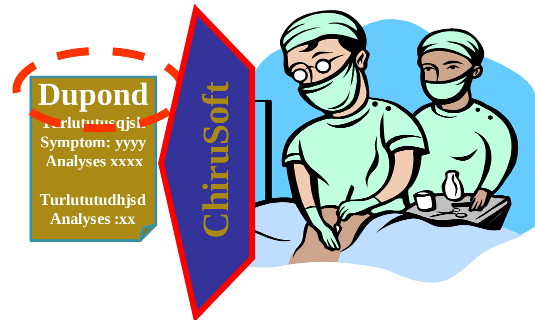
Redondance de données

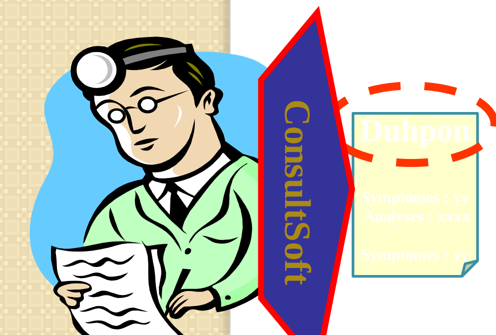
Problèmes

- ➔ **Difficultés de gestion**
- ➔ **Incohérence des données**

Interrogations









Caractéristiques

Plusieurs applications

- plusieurs formats
- plusieurs langages

Redondance de données

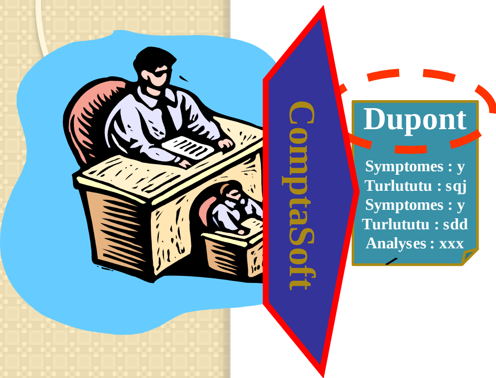
Pas de facilité d'interrogation

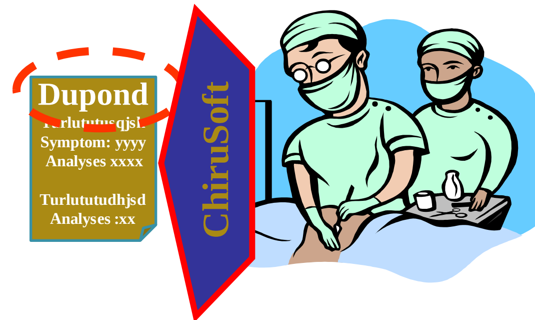
- Question $\Rightarrow$développement

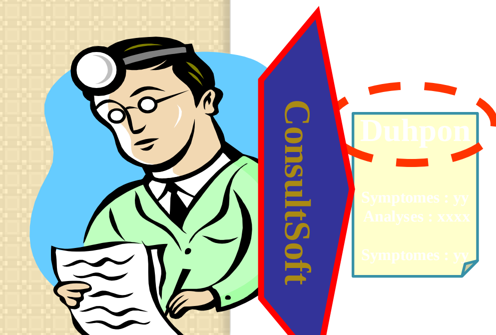
Problèmes

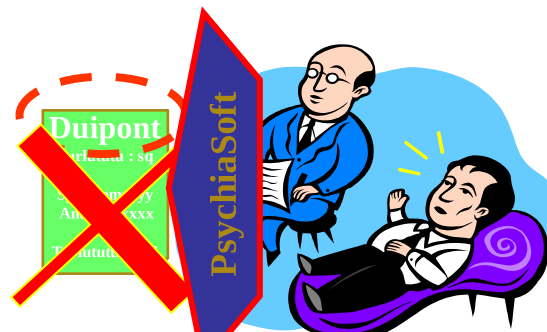
- **Difficultés de gestion**
- **Incohérence des données**
- **Coûts élevés**
- **Maintenance difficile**

Pannes ???









Caractéristiques

Plusieurs applications

- plusieurs formats
- plusieurs langages

Redondance de données

Pas de facilité d'interrogation

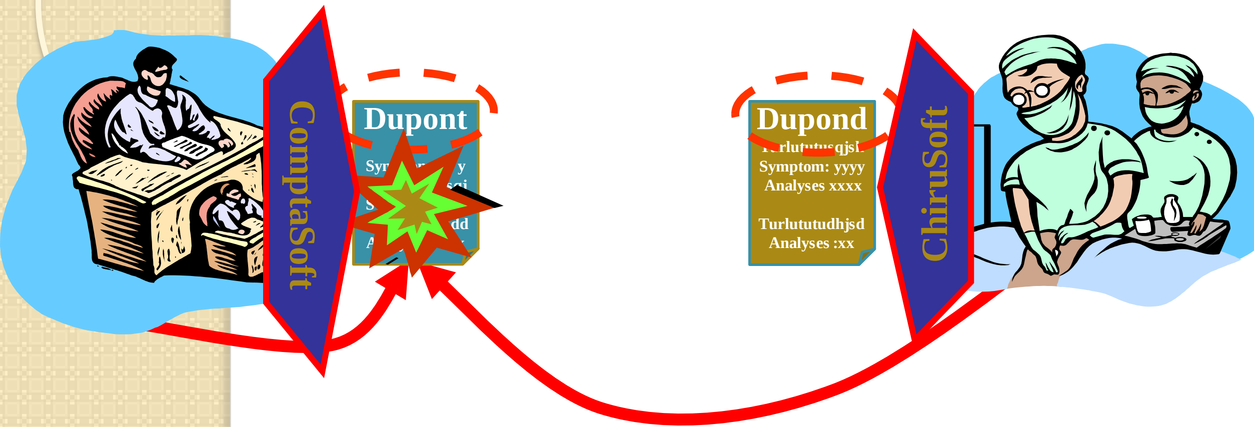
- Question ⇒développement

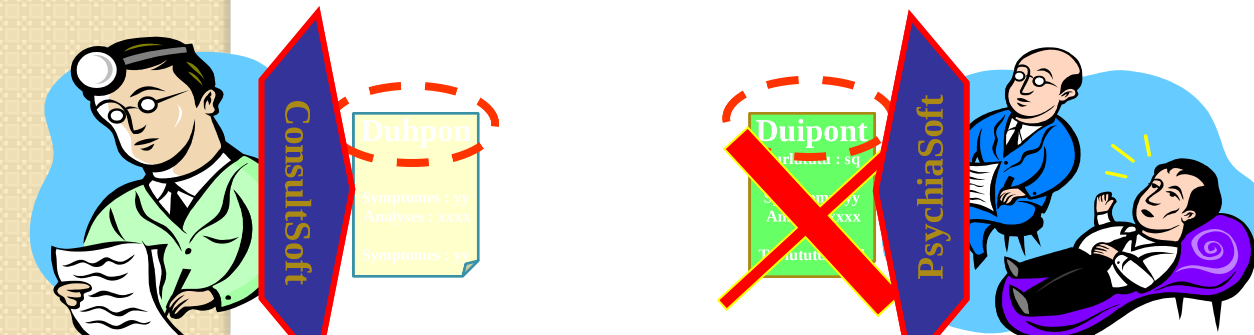
Redondance de code

Problèmes

- **Difficultés de gestion**
- **Incohérence des données**
- **Coûts élevés**
- **Maintenance difficile**
- **Gestion de pannes ???**

Partage de données





Caractéristiques

Plusieurs applications

- plusieurs formats
- plusieurs langages

Redondance de données

Pas de facilité d'interrogation

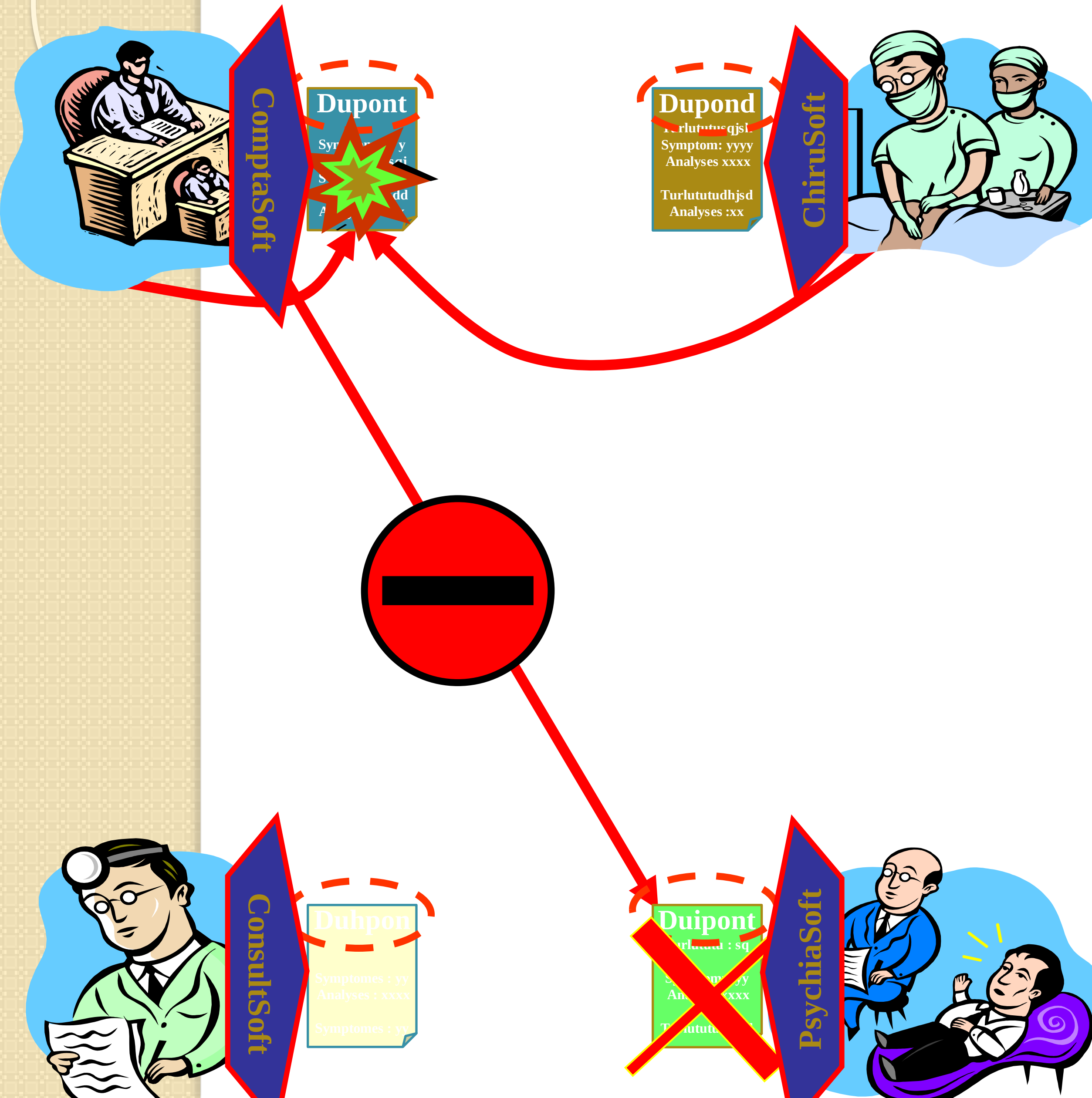
- Question ⇒développement

Redondance de code

Problèmes

- **Difficultés de gestion**
- **Incohérence des données**
- **Coûts élevés**
- **Maintenance difficile**
- **Gestion de pannes ???**
- **Partage des données ???**

Confidentialité



Caractéristiques

Plusieurs applications

- plusieurs formats
- plusieurs langages

Redondance de données

Pas de facilité d'interrogation

- Question ⇒développement

Redondance de code

Problèmes

- **Difficultés de gestion**
- **Incohérence des données**
- **Coûts élevés**
- **Maintenance difficile**
- **Gestion de pannes ???**
- **Partage des données ???**
- **Confidentialité ???**

L'approche "Bases de données"

Modélisation des données

- ➔ Eliminer la **redondance** de données
- **Centraliser** et **organiser** correctement les données
- Plusieurs niveaux de modélisation
- Outils de conception

Logiciel «Système de Gestion de Bases de Données»

- **Factorisation** des modules de contrôle des applications
  - - Interrogation, cohérence, partage, gestion de pannes, etc…
- Administration facilitées des données

Modélisation du réel

Réel					
Modèle conceptuel	• Indépendant du modèle de données • Indépendant du SGBD	Médecin — effectue — Visite			
Modèle logique	• Dépendant du modèle de données • Indépendant du SGBD	**Codasyl**	**Relationnel**	**Objet**	**XML**
Modèle Physique	• Dépendant du modèle de données • Dépendant du SGBD	• **Organisation physique des données** • **Structures de stockage des données** • **Structures accélératrices (index)**			

Modélisation Relationnelle (1)

Relation ou table

Champs, attributs, colonnes

Id-D	Nom	Prénom
1	Dupont	Pierre
2	Durand	Paul
3	Masse	Jean
….	……..	……

Tuples, lignes ou n-uplets

Modélisation Relationnelle (2)

Docteurs

Id-D	Nom	Prénom
1	Dupont	Pierre
2	Durand	Paul
3	Masse	Jean
….	……..	……

Visites

Id-D	Id-P	Id-V	Date	Prix
1	2	1	15 juin	250
1	1	2	12 août	180
2	2	3	13 juillet	350
2	3	4	1 mars	250

Prescriptions

Id-V	Ligne	Id-M	Posologie
1	1	12	1 par jour
1	2	5	10 gouttes
2	1	8	2 par jour
2	2	12	1 par jour
2	3	3	2 gouttes
….	….	….	…………

Patients

Id-P	Nom	Prénom	Ville
1	Lebeau	Jacques	Paris
2	Troger	Zoe	Evry
3	Doe	John	Paris
4	Perry	Paule	Valenton
….	…….	…….	…….

Médicaments

Id-M	Nom	Description
1	Aspegic 1000	…………………………….
2	Fluisédal	…………………………….
3	Mucomyst	…………………………….
….	……..	…………………………….

2. Objectifs des SGBD





Système de gestion de bases de données

- I- Indépendance Physique
- II- Indépendance Logique
- III – Langage de manipulation
- IV - Gestion des vues
- V - Optimisation des questions
- VI - Gestion de la cohérence
- VII - Gestion des pannes
- VIII - Concurrence d'accès
- IX - Gestion de la confidentialité
- X - Standards

BD





I - Indépendance Physique

Indépendance des programmes d'applications vis à vis du modèle physique :

Possibilité de modifier les **structures de stockage** (fichiers, index, chemins d'accès, …) sans modifier les programmes;

Ecriture des applications par des **non-spécialistes des fichiers** et des structures de stockage;

Meilleure **portabilité** des applications et **indépendance** vis à vis du matériel.

II - Indépendance Logique

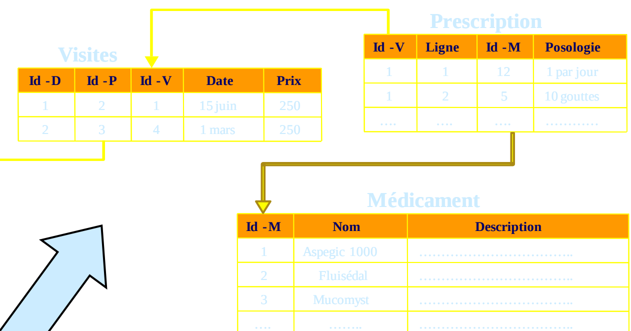
Les applications peuvent définir des **vues logiques** de la BD

Gestion des médicaments

Nombre_Médicaments

Id-M	Nom	Description	Nombre
1	Aspegic 1000	……………………………..	30
2	Fluisédal	……………………………..	20
3	Mucomyst	……………………………..	230
….	……..	……………………………..	…..

Cabinet du Dr. Masse



Visites

Id -D	Id -P	Id -V	Date	Prix
1	2	1	15 juin	250
2	3	4	1 mars	250

Prescription

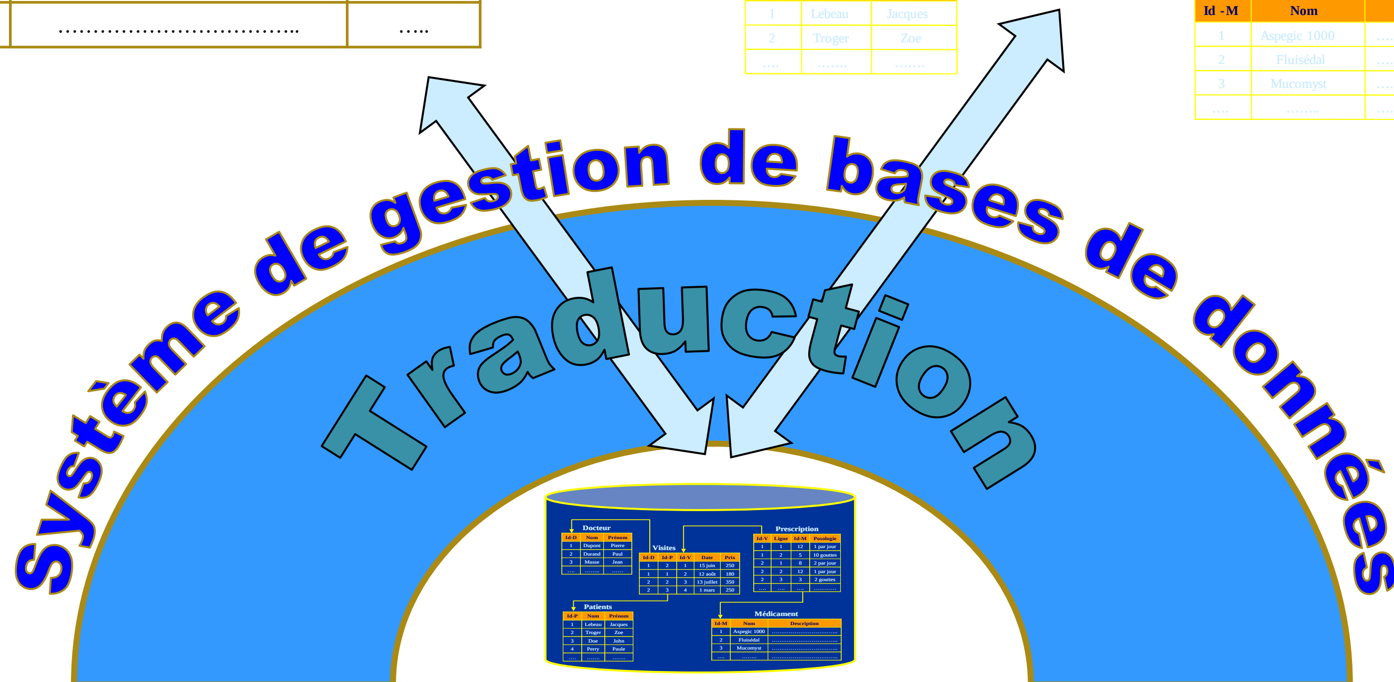
Id -V	Ligne	Id -M	Posologie
1	1	12	1 par jour
1	2	5	10 gouttes
….	….	….	…………

Patients

Id -P	Nom	Prénom
1	Lebeau	Jacques
2	Troger	Zoe
….	…….	…….

Médicament

Id -M	Nom	Description
1	Aspegic 1000	……………………………..
2	Fluisédal	……………………………..
3	Mucomyst	……………………………..
….	……..	……………………………..



Avantages de l’indépendance logique

Possibilité pour chaque application **d'ignorer** les besoins des autres (bien que partageant la même BD).

Possibilité **d'évolution de la base de données** sans réécriture des applications :

- ajout de champs, ajout de relation, renommage de champs.

Possibilité **d'intégrer des applications existantes** sans modifier les autres.

Possibilité de limiter les conséquences du partage : **Données confidentielles.**

III - Manipulation aisée

- La manipulation se fait via un langage **déclaratif**
  - La question déclare l'objectif sans décrire la méthode
  - Le langage suit une norme commune à tous les SGBD
  - **SQL : Structured Query Langage**
- Sémantique
  - Logique du 1er ordre ++
- Syntaxe (aperçu !)
  - SELECT <structure des résultats>
  - FROM <relations>
  - WHERE <conditions>

IV – Des vues multiples des données

- Les vues permettent d'implémenter l'indépendance logique en permettant de créer des **relations virtuelles**
- Vue = Question stockée
- Le SGBD stocke la **définition** et non le résultat
- Exemple :
  - la vue des patients parisiens
  - la vue des docteurs avec leurs patients
  - La vue des services statistiques
  - ...

V –Exécution et Optimisation

Traduction **automatique** des questions déclaratives en programmes procéduraux :	Optimisation **automatique** des questions	Economie de l'astuce des programmeurs
• ➔ Utilisation de l'algèbre relationnelle	• Utilisation de l'aspect déclaratif de SQL • Gestion centralisée des chemins d'accès (index, hachages, …) • Techniques d'optimisation poussées	• milliers d'heures d'écriture et de maintenance de logiciels.

VI - Intégrité Logique

- **Objectif : Détecter les mises à jour erronées**

- Contrôle sur les données élémentaires
  - Contrôle de types: ex: Nom alphabétique
  - Contrôle de valeurs: ex: Salaire mensuel entre 5 et 50kf

- Contrôle sur les relations entre les données
  - Relations entre données élémentaires:
    - Prix de vente > Prix d'achat
  - Relations entre objets:
    - Un électeur doit être inscrit sur une seule liste électorale

Contraintes d'intégrité

Avantages :

- **simplification** du code des applications
- **sécurité renforcée** par l'automatisation
- **mise en commun** des contraintes

Nécessite :

- un langage de définition de contraintes d'intégrité
- la vérification **automatique** de ces contraintes

VII - Intégrité Physique

Motivations : Tolérance aux fautes

- Transaction Failure : Contraintes d'intégrité, Annulation
- System Failure : Panne de courant, Crash serveur ...
- Media Failure : Perte du disque
- Communication Failure : Défaillance du réseau

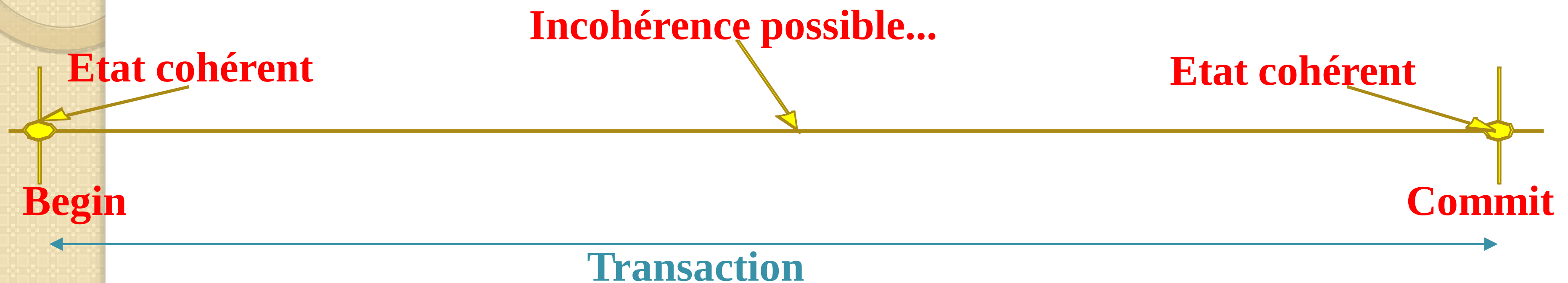
Objectifs :

- Assurer l'**atomicité** des transactions
- Garantir la **durabilité** des effets des transactions commises

Moyens :

- Journalisation : Mémorisation des états successifs des données
- Mécanismes de reprise

Transaction



Begin

CEpargne = CEpargne - 3000

CCourant = CCourant + 3000

Commit T1

Atomicité et Durabilité

ATOMICITE

Begin
CEpargne = CEpargne - 3000
CCourant = CCourant + 3000
Commit T1

Panne

➔ **Annuler le débit !!**

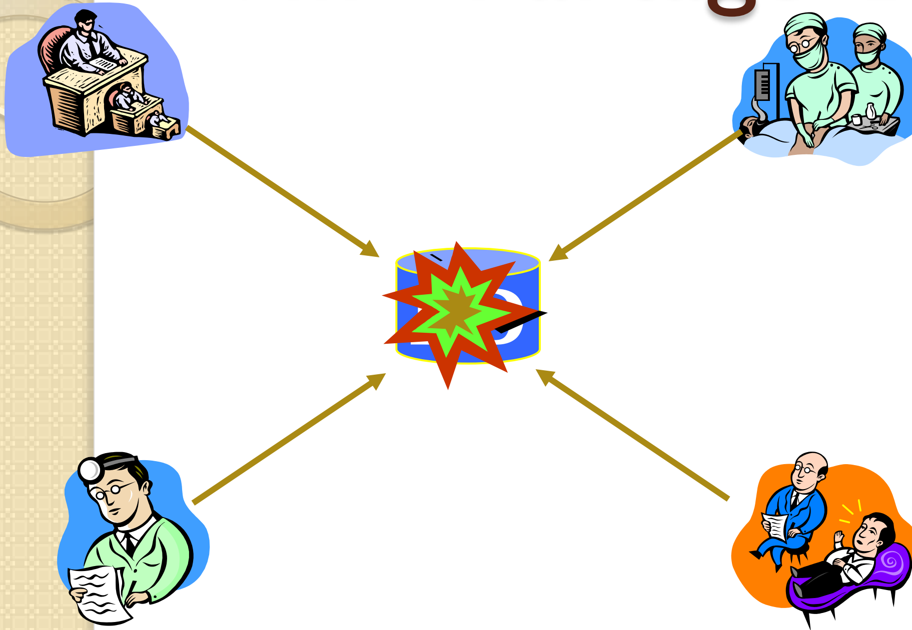
DURABILITE

Begin
CEpargne = CEpargne - 3000
CCourant = CCourant + 3000
Commit T1

Crash disque

➔ **S'assurer que le virement a été fait !**

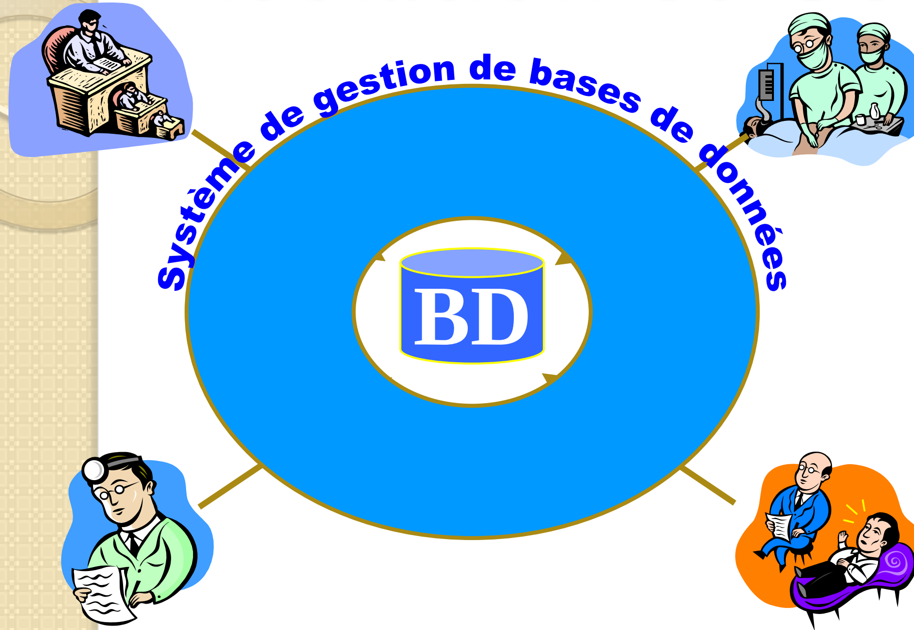
VIII - Partage des données



- Accès concurrent aux mêmes données

➔Conflits d'accès !!

Isolation et Cohérence



- Le SGBD gère les accès concurrents

➔ Chacun à l'*impression* d'être seul (Isolation)

➔ Cohérence conservée (Pas de maj conflictuelles)

IX – Confidentialité

- **Objectif : Protéger les données de la BD contre des accès non autorisés**
- Deux niveaux :
  - Connexion restreinte aux **usagers répertoriés** (mot de passe)
  - **Privilèges** d'accès aux objets de la base
- Usagers : Usager ou groupe d'usagers
- Objets : Relation, **Vue**, autres objets (procédures, etc.)

X - Standardisation

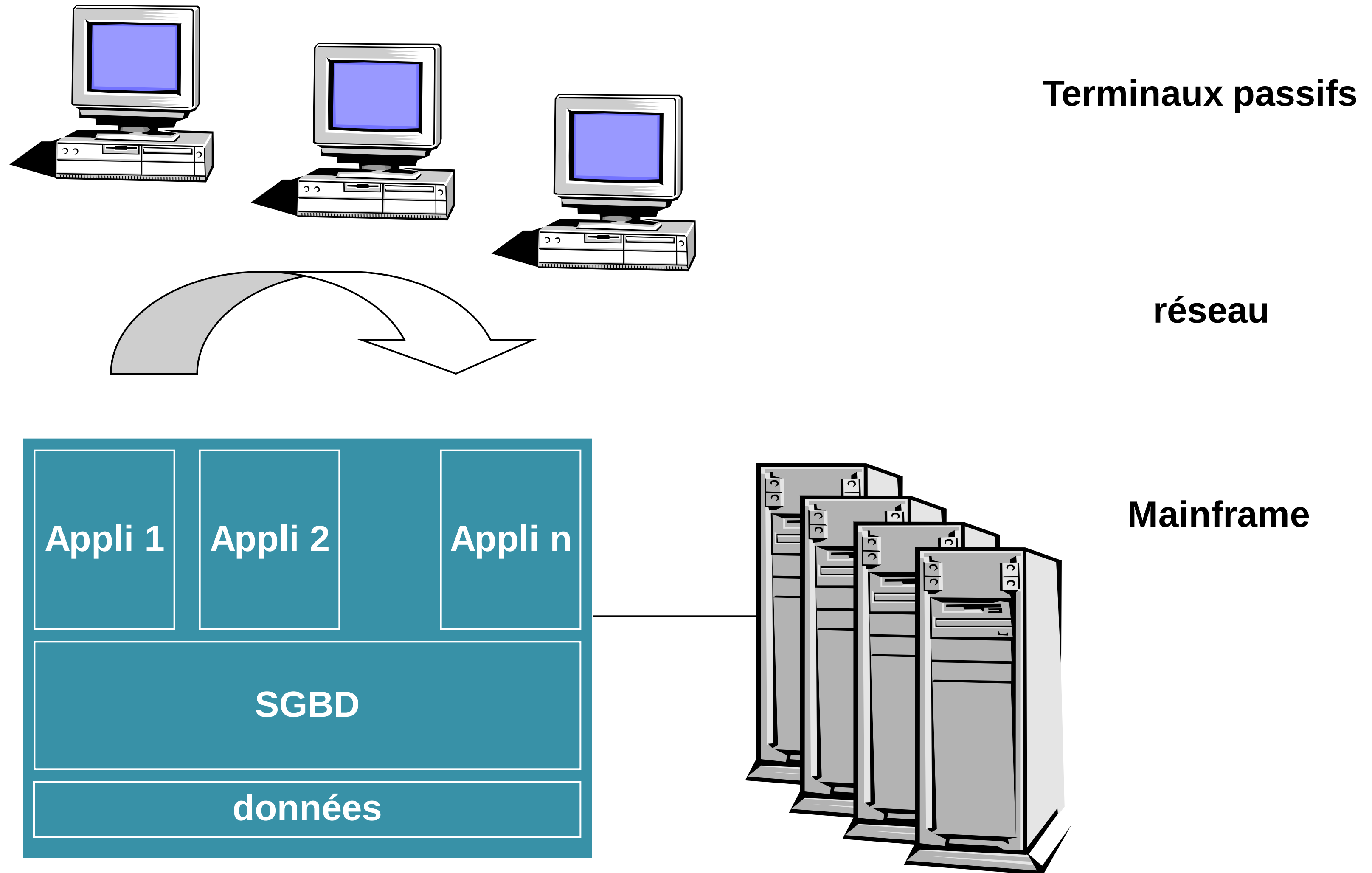
- L'approche bases de données est basée sur plusieurs standards
  - Langage SQL (SQL1, SQL2, SQL3)
  - Communication SQL CLI (ODBC / JDBC)
  - Transactions (X/Open DTP, OSI-TP)

- Force des standards
  - Portabilité
  - Interopérabilté
  - Applications multisources…

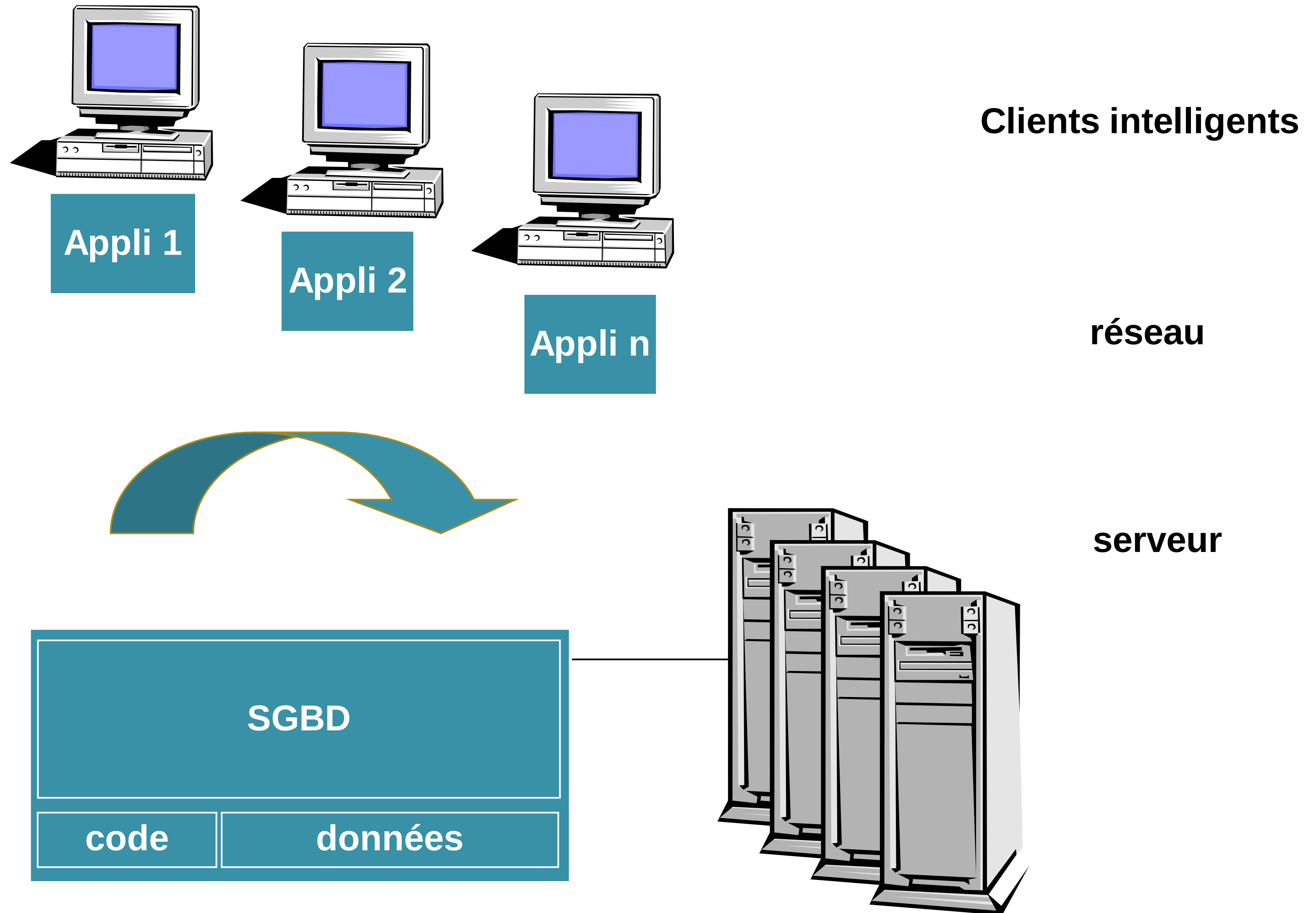
3. Architecture des SGBD

- **Les architectures physiques de SGBD sont très liées au mode de répartition.**
  - **BD centralisée**
  - **BD client/serveur**
  - **BD client/multi-serveurs**
  - **BD répartie**
  - **BD hétérogène**
  - **BD mobile**
- **Le challenge se déplace des Péta-bases aux Pico-bases.**
  - **Péta-bases => parallélisme et grandes mémoires**
  - **Pico-bases => faible empreinte et forte sécurité**

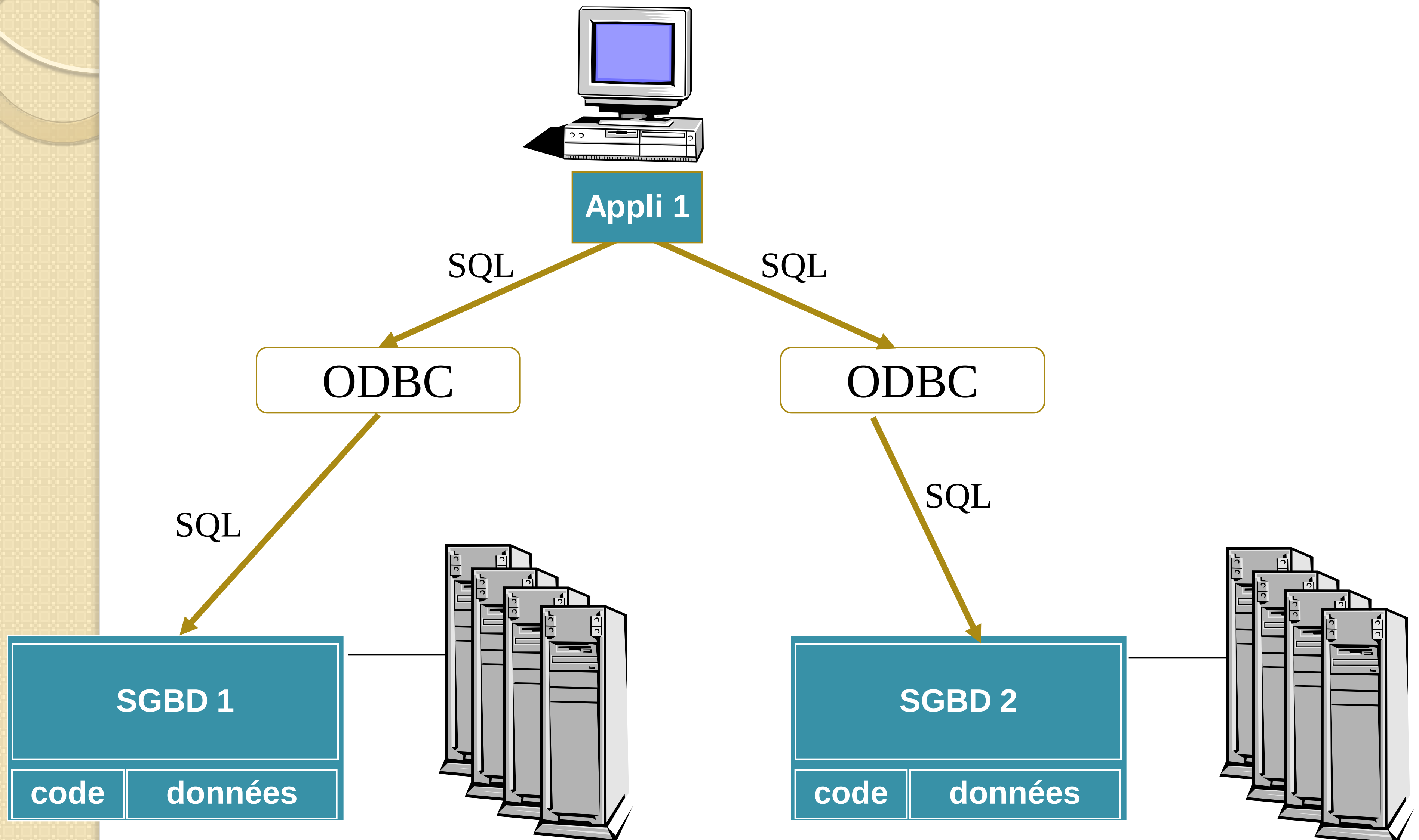
Architecture centralisée



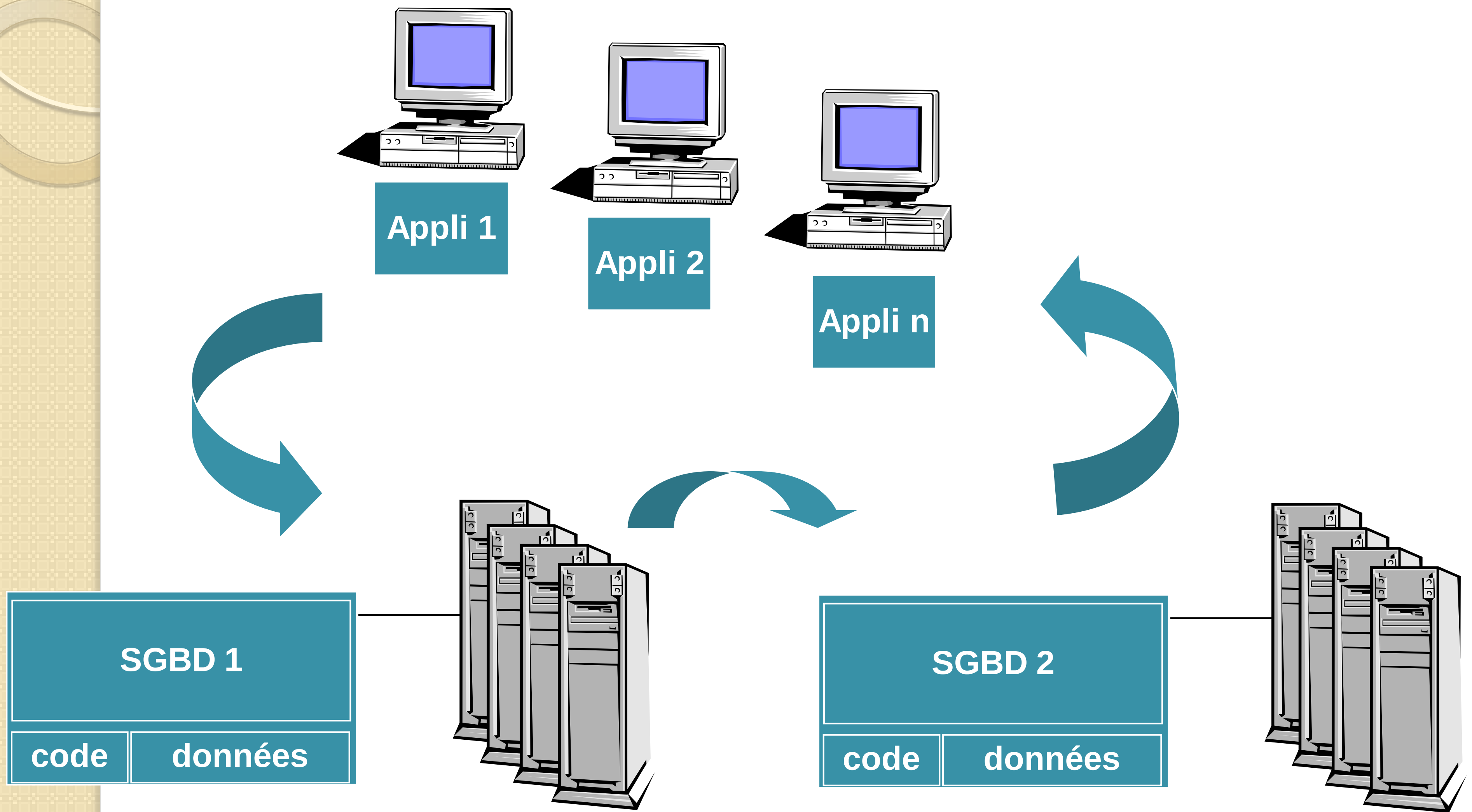
Architecture client-serveur



Architecture Client-Multiserveurs



Architecture répartie



Architecture mobile







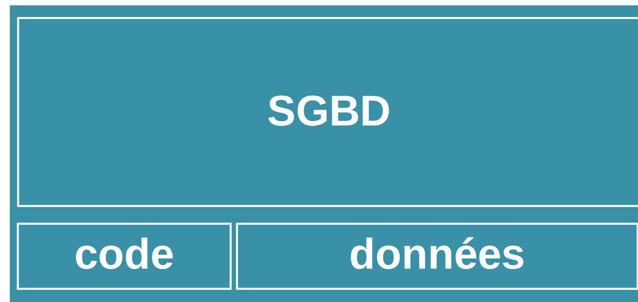
Clients intelligents mobiles

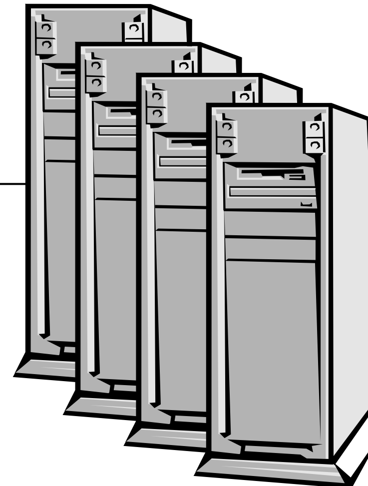
Données répliquées et/ou personnelles

Réseau sans fil



serveur





4. Applications traditionnelles des SGBD

OLTP (On Line Transaction Processing)

- Cible des SGBD depuis leur existence
- Banques, réservation en ligne ...
- Très grand nombre de transactions en parallèle
- Transactions simples

OLAP (On Line Analytical Processing)

- Entrepôts de données, DataCube, Data Mining …
- Faible nombre de transactions
- Transactions très complexes

Evolution des BD

	BD d'entreprise	BD personnelles	BD 'light' (PDA / Tél.)	PicoDBMS carte à puce
Capacité				
Prix				
Nombre				

LE MODELE RELATIONNEL

Inventé par T. Codd (IBM Recherche)

Publication ACM 1970

1. Concepts pour la description
2. Concepts pour la manipulation
3. Concepts additionnels

1. CONCEPTS DESCRIPTIFS

- Ensemble de concepts pour formaliser la description d'articles de fichiers plats

- Modèle standardisé mais extensible
  - Introduction de types de données variés (SQL2)
  - Introduction de la dynamique (produits, SQL3)
  - Introduction des objets (SQL3)

Domaine

- ENSEMBLE DE VALEURS
- Exemples:
  - ENTIER
  - REEL
  - CHAINES DE CARACTERES
  - EUROS
  - SALAIRE = {4 000..100 000}
  - COULEUR= {BLEU, BLANC, ROUGE}

Produit cartésien

- LE PRODUIT CARTESIEN D1x D2x ... x Dn EST L'ENSEMBLE DES TUPLES (N-UPLETS) <V1,V2,....Vn> TELS QUE Vi ∈ Di
- Exemple:
  - D1 = {Bleu,Blanc,Rouge}
  - D2 = {Vrai, Faux}

Bleu	Vrai
Bleu	Faux
Blanc	Vrai
Blanc	Faux
Rouge	Vrai
Rouge	Faux

Relation

- SOUS-ENSEMBLE DU PRODUIT CARTESIEN D'UNE LISTE DE DOMAINES

- Une relation est caractérisée par un nom

- Exemple:
  - D1 = COULEUR
  - D2 = BOOLEEN

Coulvoiture	Coul	Choix
	Bleu	Faux
	Blanc	Vrai
	Rouge	Vrai

Attribut

- VISION TABULAIRE DU RELATIONNEL
  - Une relation est une table à deux dimensions
  - Une ligne est un tuple
  - Un nom est associé à chaque colonne afin de la repérer indépendamment de son numéro d'ordre

- ATTRIBUT
  - nom donné à une colonne d'une relation
  - prend ses valeurs dans un domaine

Clé

- GROUPE D'ATTRIBUTS MINIMUM QUI DETERMINE UN TUPLE UNIQUE DANS UNE RELATION
- Exemples:
  - NSS DANS PERSONNE
- CONTRAINTE D'ENTITE
  - Toute relation doit posséder au moins une clé documentée

Schéma

- NOM DE LA RELATION, LISTE DES ATTRIBUTS AVEC DOMAINES, ET LISTE DES CLES D'UNE RELATION
- Exemple:
  - voiture(id: Int, nom:texte, annee Cons:entier)
  - Par convention, la clé primaire est soulignée
- SCHEMA D'UNE BD RELATIONNELLE
  - C'est l'ensemble des schémas des relations composantes

Clé Etrangère

- GROUPE D'ATTRIBUTS DEVANT APPARAITRE COMME CLE DANS UNE AUTRE RELATION
- Les clés étrangères définissent les contraintes d'intégrité référentielles
  - Lors d'une insertion, la valeur des attributs doit exister dans la relation référencée
  - Lors d'une suppression dans la relation référencée les tuples référençant doivent disparaître
  - Elles correspondent aux liens entité-association obligatoires

Exemple de Schéma

- EXEMPLE

  personnes (<u>NB</u>, NOM, PRENOM, TYPE)

  the (<u>Nthe</u>, nom, couleur, origine)

  consommation(NB, Nthe, DATE, QUANTITE)

- CLES ETRANGERES

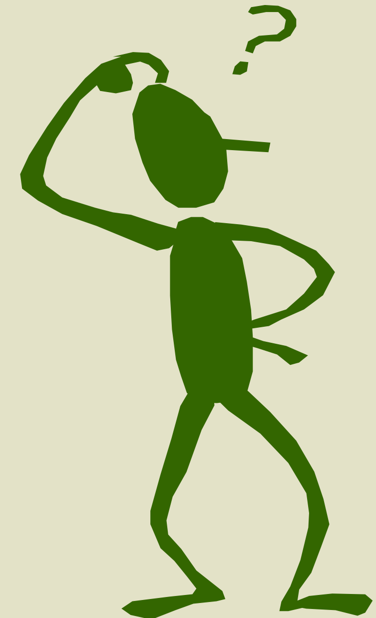
  consommation.Nthe REFERENCE the.Nthe

  consmmation.NB REFERENCE personne.NB

Concepts Descriptifs : Bilan

- RELATION ou TABLE
- ATTRIBUT ou COLONNE
- DOMAINE ou TYPE
- CLE
- CLE ETRANGERE

Questions ?



Synthèse : Create Table

- CREATION DES TABLES EN SQL
  - CREATE TABLE <relation name>
  - (<attribute definition>+)
  - [{PRIMARY KEY | UNIQUE} (<attribute name>+)]
- avec :
  - <attribute definition> ::= <attribute name> <data type>
  - [NOT NULL [{UNIQUE | PRIMARY KEY}]]
- Exemple :
  - CREATE TABLE the
  - (Nthe INTEGER PRIMARY KEY
  - Nom VARCHAR
  - ….

2. CONCEPTS MANIPULATOIRES

- Un ensemble d'opérations formelles
  - Algèbre relationnelle
- Ces opérations permettent d'exprimer toutes les requêtes sous forme d'expressions algébriques
- Elles sont la base du langage SQL
  - Paraphrasage en anglais des expressions relationnelles
- Ces opérations se généralisent à l'objet
  - Algèbre d'objets complexes

Opérations Ensemblistes

- UNION   notée $\cup$
- Produit cartésien   notée X
- DIFFERENCE notée —

Opérations Spécifiques

Projection

- Elimination des attributs non désirés et suppression des tuples en double
- Relation -> Relation notée:

  $\pi_{A1,A2,...Ap}$ (R)

personne	prenom	dateNaiss	villeNaiss
ahmed		1983	casablanca
sara		1979	rabat
khalid		1983	oujda
Amina		1986	oujda

πnom,dateNai

π(personne)	nom	dateNaiss
	ahmed	1983
	sara	1979
	khalid	1983
	Amina	1986

Restriction

- Obtention des tuples de R satisfaisant un critère Q
- Relation ->Relation, notée $\sigma_Q(R)$
- Q est le critère de qualification de la forme :
  - Ai θ Valeur
  - θ = □ { =, <, >=, >, <=, !=}

Exemple de Restriction

$\sigma_{\text{dateNaiss}>1983}$



personne	prenom	dateNaiss	villeNaiss
	Amina	1986	oujda

Jointure

- Composition des deux relations sur un domaine commun
- Relation X Relation ->Relation
  - notée
- Critère de jointure $\bowtie$
  - Attributs de même nom égaux :
    - Attribut = Attribut
    - Jointure naturelle
  - Comparaison d'attributs :
    - Attribut1 Θ Attribut2
    - Théta-jointure

SQL

- Une requête SQL est un paraphrasage d'une expression de l'algèbre relationnelle en anglais
- Requête élémentaire :

```
SELECT A1, A2, …Ap
FROM R1, R2, …Rk
WHERE Q   [{UNION |INTERSECT | EXCEPT } … ]
```

- Sémantique du bloc select :

```
PROJECT A1,A2,…Ap (
          RESTRICT Q  (
                   PRODUIT ( R1, R2, …, Rk) ) )
```

3. CONCEPTS ADDITIONNELS

- Ensemble de concepts pour :
  - Etendre les fonctionnalités de manipulation
  - Décrire les règles d'évolution des données
  - Supporter des objets complexes (SQL3)

- Introduits progressivement dans le modèle :
  - Complique parfois le modèle
  - Standardisés au niveau de SQL3 (1999)
  - Des extensions multiples …

Fonction et Agrégat

- FONCTION
  - Fonction de calcul en ligne appliquée sur un ou plusieurs attributs

- AGREGAT
  - Partitionnement horizontal d'une relation selon les valeurs d'un groupe d'attributs, suivi d'un regroupement par une fonction de calcul en colonne (SUM, MIN, MAX, AVG, COUNT, …)

4. CONCLUSION

- Un ensemble de concepts bien compris et bien formalisés
- Un modèle unique, riche et standardisé
  - intégration des BD actives
  - intégration des BD objets
- Un formalisme qui s'étend plutôt bien
  - algèbre d'objets
- Un langage associé défini à plusieurs niveaux
  - SQL1, 2, 3

LE LANGAGE DE REQUETES SQL INTRODUCTION

- Origines et Evolutions
- SQL1 86: la base
- SQL1 89: l'intégrité

1. Origines et Evolutions

- SQL est dérivé de l'algèbre relationnelle et de SEQUEL
- Il a été intégré à DB2, ORACLE, MySQL, SQLServer, etc.
- Il existe trois versions normalisées, du simple au complexe :
  - SQL1 86 version minimale
  - SQL1 89 (intégrité)
  - SQL2 (92) langage complet
- Une version 3 étendue (objets, règles) est la norme 99.
- La plupart des systèmes supportent SQL2 complet

Opérations

- Opérations de base
  - SELECT, INSERT, UPDATE, DELETE
- Opérations additionnelles
  - définition et modification de schémas
  - définition de contraintes d'intégrité
  - définition de vues
  - accord des autorisations
  - gestion de transactions

Organisation du Langage

- SQL comprend quatre parties :
- Le langage de définition de schéma (Tables, Vues, Droits)
- Le langage de manipulation (Sélection et mises à jour)
- La spécification de modules appelables (Procédures)
- L'intégration aux langages de programmation

SQL1 - 86

- LANGAGE DE DEFINITIONS DE DONNEES
  - CREATE TABLE
  - CREATE VIEW
- LANGAGE DE MANIPULATION DE DONNEES
  - SELECT OPEN
  - INSERT FETCH
  - UPDATE CLOSE
  - DELETE
- LANGAGE DE CONTROLE DE DONNEES
  - GRANT et REVOKE
  - BEGIN et END TRANSACTION
  - COMMIT et ROLLBACK

2. SELECT: Forme Générale

- SELECT <liste de projection>
- FROM <liste de tables>
- [WHERE <critère de jointure> AND <critère de restriction>]
- [GROUP BY <attributs de partitionnement>]
- [HAVING <citère de restriction>]

◆ Restriction :

- arithmétique (=, <, >, ≠, ≥,≤□)
- textuelle (LIKE)
- sur intervalle (BETWEEN)
- sur liste (IN)

◆ **Possibilité de blocs imbriqués par :**

- **IN, EXISTS, NOT EXISTS, ALL, SOME, ANY**

Forme générale de la condition

<**search condition**> ::= **[**NOT**]**

<**nom_colonne**> θ **constante** | <**nom_colonne**>

<**nom_colonne**> LIKE **<modèle_de_chaîne>**

<**nom_colonne**> IN <**liste_de_valeurs**>

<**nom_colonne**> **θ (**ALL | ANY | SOME**)** <**liste_de_valeurs**>

EXISTS <**liste_de_valeurs**>

UNIQUE <**liste_de_valeurs**>

<**tuple**> MATCH **[**UNIQUE**]** <**liste_de_tuples**>

<**nom_colonne**> BETWEEN **constante** AND **constante**

<**search condition**> AND | OR <**search condition**>

avec

θ ::= < | = | > | ≥ | ≤ | <>

Remarque: <**liste_de_valeurs**> **peut être dynamiquement déterminée par une requête**

3. Les Mises à Jour

- INSERT
  - Insertion de lignes dans une table
  - Via formulaire où via requêtes
- UPDATE
  - Modification de lignes dans une table
- DELETE
  - Modification de lignes dans une table

Commande INSERT

- INSERT INTO <relation name>
  [(attribute [,attribute] …)]
  {VALUES <value spec.> [, <value spec.>] …| <query spec.>}

Commande UPDATE

UPDATE <relation name>

SET <attribute = {value expression | NULL}

[<attribute> = {value expression | NULL}] …

[WHERE <search condition>]

Commande DELETE

- DELETE FROM <relation name>
- [WHERE <search condition>]

4. Contraintes d'intégrité

- Contraintes de domaine
  - Valeurs possibles pour une colonne
- Contraintes de clés primaires
  - Clé et unicité
- Contraintes référentielles(clé étrangères)
  - Définition des liens inter-tables

SQL1 - 89 : INTEGRITE

- VALEURS PAR DEFAUT
  - CREATE TABLE The
  - (Nthe INT UNIQUE,
  - origine CHAR(10),
  - annee INT,
  -)
- CONTRAINTES DE DOMAINES
  - SALAIRE INT CHECK BETWEEN 6000 AND 100000

SQL1 - 89 : Contrainte référentielle

- Clé primaire et contrainte référentielle
  - A discuter

SQL1 – 89 : Création de table

CREATE TABLE <nom_table>
(<def_colonne> *
[<def_contrainte_table>*]) ;

< def_colonne > ::=
<nom_colonne> < type | nom_domaine >
[CONSTRAINT nom_contrainte
< NOT NULL | UNIQUE | PRIMARY KEY |
CHECK (condition) | REFERENCES nom_table (liste_colonnes) >]

< def_contrainte_table > ::= CONSTRAINT nom_contrainte
< UNIQUE (liste_colonnes) | PRIMARY KEY (liste_colonnes) |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) REFERENCES nom_table (liste_colonnes) >
[NOT] DEFERRABLE

Autre création de tables

```
CREATE TABLE EXPEDITIONS
  ( numExp INTEGER PRIMARY KEY
    date_exp DATE,
    qte QUANTITE,
    CONSTRAINT refCom FOREIGN KEY numExp
       REFERENCES  COMMANDES (numCom) DEFERRABLE
) ;
```

L'association d'un nom à une contrainte est optionnelle.

Ce nom peut être utilisé pour référencer la contrainte (ex: messages d'erreurs).

5. CONCLUSION

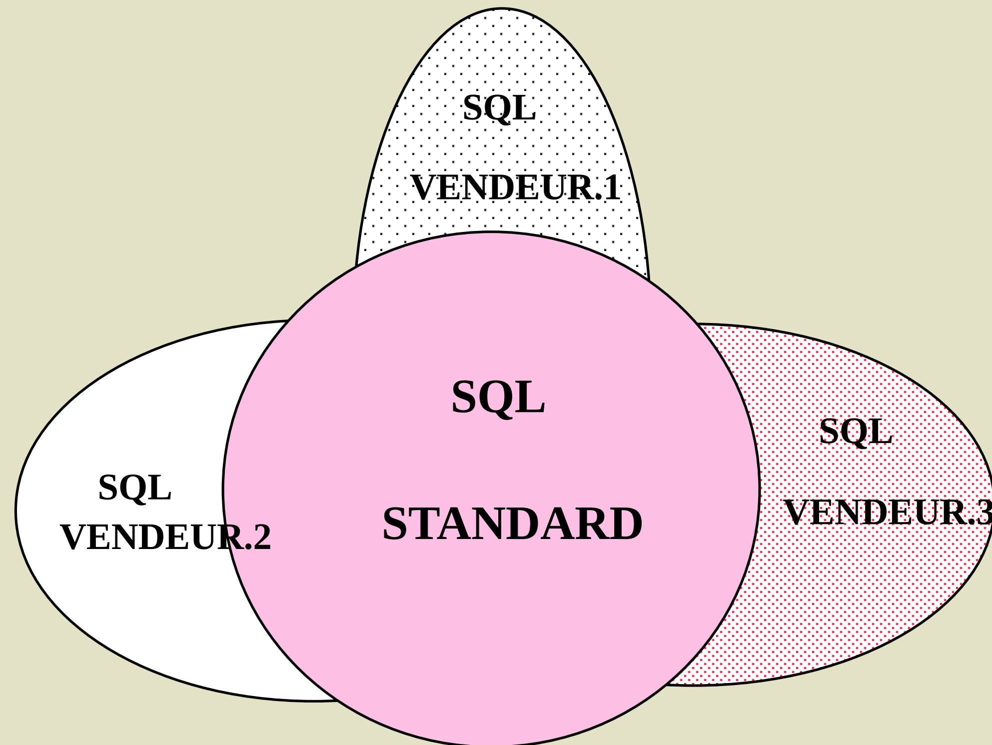
- SQL1 est un standard minimum
- Les versions étendues:
  - SQL2 = Complétude relationnelle
  - SQL3 = Support de l'objet
- Sont aujourd'hui intégrées dans les grands SGBD

LA NORMALISATION DE SQL

- Groupe de travail ANSI/X3/H2 et ISO/IEC JTC1/SC2
- Documents ISO :
  - SQL1 - 86 : Database Language SQL X3.135 ISO-9075-1987)
  - SQL1 - 89 : Database Language SQL with Integrity Enhancement X3.168 ISO-9075-1989
  - SQL2 - 92 : Database Language SQL2 X3.135 ISO-9075-1992
- Arguments pour :
  - Réduction des coûts d'apprentissage
  - Portabilité des applications
  - Longévité des applications
  - Langage de communication inter-systèmes
- Arguments contre :
  - Manque de rigueur théorique
  - Affaiblit la créativité

POSITION DES VENDEURS

- Problèmes !!!



SQL
EXTRACTION DE DONNÉES

Vue d'ensemble

- Extraction de données à l'aide de l'instruction SELECT
- Filtrage des données (Restriction)
- Mise en forme des ensembles de résultats

Extraction de données: utilisation de l'instruction SELECT

- La liste de sélection indique les colonnes
- La clause WHERE indique la condition limitant la requete
- La clause FROM indique la table

Extraction de données: Spécification des colonnes

- SELECT emp_ID, nom, prenom
- FROM employe

Extraction de données: utilisation de la clause WHERE pour specifier des lignes

- SELECT emp_ID, nom
- FROM employe
- WHERE empl_ID=5

Filtrage des données

- Utilisation des opérateurs de comparaison
- Utilisation des comparaisons de chaines
- Utilisation des opérateurs logiques
- Extraction d'une plage de valeurs
- Utilisation d'une liste de valeurs comme critère de recherche
- Extraction de valeurs inconnues

Filtrage des données

Type de filtre	Condition de recherche
Opérateurs de comparaison	=,<,>,<=,>= et <>
Comparaison de chaines	LIKE et not LIKE
Opérateurs logiques	AND, OR, NOT
Plage de valeurs	BETWEEN et NOT BETWEEN
Liste de valeurs	IN et NOT IN
Valeur inconnues	IS NULL et IS NOT NULL

Filtrage des données: Utilisation des opérateurs de comparaison

- SELECT prenom, ville
- FROM employe
- WHERE pays=‘MAROC’

Filtrage des données: Utilisation des comparaison de chaines

- SELECT nom_entreprise
- FROM client
- WHERE nom_entreprise LIKE '%MAROC%'

Filtrage des données: Utilisation des opérateurs logiques

- SELECT produit_ID, produit_nom
- FROM produit
- WHERE (produit_nom LIKE ‘toto’ OR produit_ID=20) AND (PU_HT>100)

Filtrage des données: Extraction d'une plage de valeurs

- SELECT produit_nom, PU_HT
- FROM produit
- WHERE PU_HT BETWEEN 10 AND 20

Filtrage des données: Utilisation d’une liste de valeurs comme critère de recherche

- SELECT nom_entreprise, pays
- FROM fournisseurs
- WHERE pays IN (‘MAROC’,’ALGERIE’)

Filtrage des données: Extraction de valeurs NULL

- SELECT nom_entreprise, fax
- FROM fournisseurs
- WHERE fax IS NULL

Mise en forme des ensembles de résultats

- Tri des données
- Suppression des doublons
- Changement des noms de colonne
- Utilisation de litteraux

Mise en forme :
Tri des données

- SELECT produit_ID, nom_produit, categorie_ID
- FROM produit
- ORDER BY categorie_ID

Mise en forme : Suppression des doublons

- SELECT DISTINCT pays
- FROM fournisseurs
- ORDER BY PAYS

Mise en forme : Changement des noms de colonne

- SELECT nom as N, prenom as P, employe_ID as ‘EMPLOYE ID’
- FROM employe

Mise en forme : Utilisation de litteraux

- SELECT nom, ‘NUMERO IDENTIFICATION :’ employe_ID
- FROM employe

SQL

Regroupement

Utilisation de fonctions d’agrégation

Fonction d’agrégation	Desciption
AVG (expr)	Moyenne de expr
COUNT (* | expr)	Nombre de ligne
MAX (expr)	Maximum de expr
MIN (expr)	Minimum de expr
SUM (expr)	Somme de exp
STDEV (expr)	Ecart type de exp
VAR IANCE(expr)	Variance de exp

Utilisation de fonctions d'agrégation

- Exemple

Select SUM(Quantite)
From detail_commande

Utilisation de la clause GROUP BY

- Select produit_ID, Commande_ID, Quantite
- From detail_commande

- Select produit, SUM(Quantite)
- From detail_commande
- GROUP BY produit_ID

Utilisation de la clause GROUP BY

Produit_ID	Commande_ID	Quantite
1	1	5
1	4	10
1	7	20
2	1	5
2	2	10



Produit_ID	Total_Qte
1	35
2	15

GROUP BY avec Having

- SELECT produit_ID, SUM(Quantite),
- From detail_commande
- GROUP BY produit_ID
- HAVING SUM(Quantite) >30

MANIPULATION DE DONNÉES

Objectif

Décrire l'aspect LMD (langage de manipulation des données) de MySQL. Nous verrons que SQL propose trois instructions pour manipuler des données :

- l'insertion d'enregistrements : INSERT ;
- la modification de données : UPDATE ;
- la suppression d'enregistrements : DELETE (et TRUNCATE).

Il existe d'autres possibilités pour insérer des

données en utilisant des outils d'importation ou de migration, citons MySQL Migration Toolkit, SQLPorter, Navicat, Intelligent Converters et MySQL Data Import de la société EMS.

Insertions d’enregistrements (INSERT)

Insertions d’enregistrements (INSERT)

- Pour pouvoir insérer des enregistrements dans une table, il faut que vous ayez reçu le privilège INSERT. Il existe plusieurs possibilités d’insertion : l’insertion monoligne qui ajoute un enregistrement par instruction et l’insertion multiligne qui insère plusieurs enregistrements par une requête.

Renseigner toutes les colonnes

```
INSERT INTO Compagnie
VALUES ('SING', 7, 'Camparols', 'Singapour',
'Singapore AL');
```

Renseigner certaines colonnes

- Insérons deux lignes dans la table Compagnie en ne précisant pas toutes les colonnes. La première insertion affecte implicitement la valeur par défaut à la colonne ville. La deuxième donne explicitement la valeur NULL à la colonne nrue.

```
INSERT INTO Compagnie
VALUES ('AC', 10, 'Gambetta', DEFAULT,
'Air France');

INSERT INTO Compagnie(comp, nrue, rue,
nomComp)
VALUES ('AF', NULL, 'Champs Elysées', 'Castanet
Air');
```

Plusieurs enregistrements

Le script suivant ajoute trois nouvelles compagnies en une seule instruction INSERT.

```
INSERT INTO Compagnie VALUES
('LUFT',9,'Salas','Munich','Luftansa'),
('QUAN',1,'Kangouroo','Sydney','Quantas'),
('SNCM',3,'P. Paoli','Bastia','Corse Air');
```

Énumérations

Le type ENUM est considéré comme une liste de chaînes de caractères. Toute valeur d’unecolonne de ce type devra appartenir à cette liste établie lors de la création de la table. Supposons qu’on recense quatre types possibles de diplômes ('BTS', 'DUT', 'Licence' et 'INSA') pour chaque étudiant. On ne stocke qu’un seul diplôme par étudiant.

```
CREATE TABLE UnCursus
(num CHAR(4), nom CHAR(15), diplome
ENUM ('BTS','DUT','Licence','INSA') ,
CONSTRAINT pk_Cusus PRIMARY KEY(num));

INSERT INTO UnCursus VALUES
('E1', 'F. Brouard', ('BTS'));

INSERT INTO UnCursus VALUES
('E2', 'F. Degrelle', 'Licence');
```

Dates et heures

Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures, années, et heures). Les fonctions NOW() et SYSDATE() retournent la date et l'heure courantes. Dans une procédure ou un déclencheur SYSDATE est réévaluée en temps réel, alors que NOW désignera toujours l'instant de début de traitement

Dates et heures

type	commentaire
DATE	Sur 3 octets. L'affichage est au format 'YYYY-MM-DD'.
DATETIME	Sur 8 octets. L'affichage est au format 'YYYY-MM-DD HH:MM:SS'.
YEAR	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est 'YYYY'.
TIME	L'heure au format 'HHH:MM:SS' sur 3 octets.
TIMESTAMP	Estampille sur 4 octets (au format 'YYYY-MM-DD HH:MM:SS') ; mise à jour à chaque modification sur la table.

Dates et heures

CREATE TABLE Pilote

(brevet VARCHAR(6), nom VARCHAR(20), dateNaiss **DATE**,

nbHVol DECIMAL(7,2),dateEmbauche **DATETIME**, compa VARCHAR(4),

CONSTRAINT pk_Pilote PRIMARY KEY(brevet));

Modifications de colonnes UPDATE

Modifications de colonnes UPDATE

L'instruction UPDATE permet la mise à jour des colonnes d'une table. Pour pouvoir modifier des enregistrements d'une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège UPDATE sur la table.

Modification d'une colonne

Exemple :

Modifions la compagnie de code 'AN1' en affectant la valeur 50 à la colonne nrue.

```
UPDATE Compagnie SET nrue = 50 WHERE
comp = 'AN1';
```

Modification de plusieurs colonnes

- Exemple:

UPDATE Compagnie SET nrue = 14, ville = **DEFAULT** WHERE comp = 'AN2';

UPDATE Pilote SET dateNaiss = '1967-03-25 **12:35:00'**
WHERE brevet = 'PL-1'

Suppressions d’enregistrements

Instruction DELETE

Les instructions DELETE permet de supprimer un ou plusieurs enregistrements d'une table. Pour pouvoir supprimer des enregistrements dans une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège DELETE sur la table.

DELETE FROM Pilote WHERE compa = 'AF';

DELETE FROM Compagnie WHERE comp = 'AF';

Intégrité référentielle

Intégrité référentielle

L'intégrité référentielle forme le cœur de la cohérence d'une base de données relationnelle. Cette intégrité est fondée sur la relation entre clés étrangères et clés primaires (ou candidates : colonnes indexées uniques et non nulles) qui permettent de programmer des règles de gestion.

Ce faisant, la plupart des contrôles côté client (interface) sont ainsi déportés côté serveur.

!!!! A continuer…

JOINTURE (SUITE)

Sous-requêtes

Jointure procédurale (Sous requêtes)

- Les jointures procédurales sont écrites par des requêtes qui contiennent des sous-interrogations (SELECT imbriqué). Chaque clause FROM ne contient qu'une seule table.

Jointure procédurale

SELECT *colonnesTable1 FROM [nomBase.]nomTable1*

WHERE *colonne(s)* | *expression(s) { IN* | = | *opérateur }*

(SELECT *colonne(s)delaTable2 FROM [nomBase.]nomTable2*

WHERE *colonne(s)* | *expression(s) { IN* | = | *opérateur }*

(SELECT …)

[AND (*conditionsTable2)]*

)

[AND (*conditionsTable1)];*

Jointures procédurales

*Sous-interrogations **monolignes***

Voir Exemples!!

Jointures procédurales

Sous-interrogations ***Multilignes***

Les opérateurs multilignes sont les suivants (1/2) :

1. IN compare un élément à une donnée quelconque d'une liste ramenée par la sous-interrogation. Cet opérateur est utilisé pour les équijointures et les autojointures.
2. ANY compare l'élément à chaque donnée ramenée par la sous-interrogation. L'opérateur « =ANY » équivaut à IN. L'opérateur « <ANY » signifie « inférieur à au moins une des valeurs. L'opérateur « >ANY » signifie « supérieur à au moins une des valeurs ».

Les opérateurs multilignes sont les suivants (1/2) :

3. ALL compare l'élément à tous ceux ramenés par la sous-interrogation. L'opérateur « <ALL » signifie « inférieur au minimum » et « >ALL » signifie « supérieur au maximum ».

Voir Exemples!!

Sous-interrogations synchronisées

Sous-interrogations synchronisées

SELECT ***alias1.c***
FROM *nomTable1* ***alias1***
WHERE *colonne(s) opérateur*
(SELECT alias2.z…
FROM *nomTable2 alias2*
WHERE ***alias2 .x opérateur alias1.y)***
[AND (*conditionsTable1)];*

Sous-interrogations synchronisées: les opérateurs EXISTS & NOT EXISTS

- L’opérateur EXISTS permet d’interrompre la sous-interrogation dès le premier enregistrement trouvé. La valeur FALSE est retournée si aucun enregistrement n’est extrait par la sous-interrogation
- L’opérateur NOT EXISTS retourne la valeur TRUE si aucun enregistrement n’est extrait par la sous-interrogation.

Voir Exemples!!

Question?

Exemples de jointures

Notre premier exemple

Compagnie

Comp	nrue	rue	ville	nomComp
AF	122	Paris	Paris	Air France
RAM	7	BD des FAR	Casablanca	Royal Air Maroc
SING	1	Camparols	Singapour	Singapore AL

Pilote

brevet	nom	nbVol	compa	chefPil
PL1	Pierre lamo	450	AF	PL4
PL2	Didier Linx	900	AF	PL4
PL3	Hamid Lmrabet	1000	RAM	
PL4	Henri alqui	3400	AF	
PL5	Ahmed Saadi	950	RAM	

Deux tables à mettre en jointure

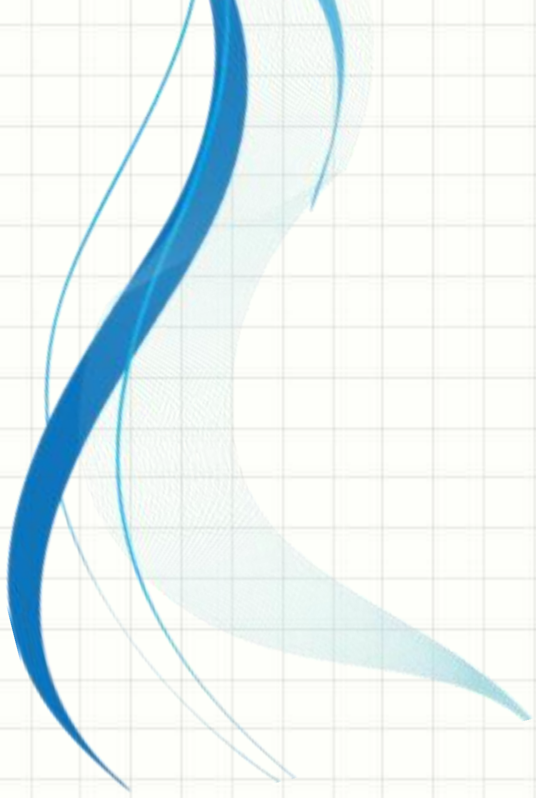
Comp	nrue	rue	ville	nomComp
AF	122	Paris	Paris	Air France
RAM		BD des FAR	Casablanca	Royal Air Maroc
SING	1	Camparols	Singapour	Singapore AL

brevet	nom	nbVol	compa	chefPil
PL1	Pierre lamo		AF	PL4
PL2	Didier Linx	900		PL4
PL3	Hamid Lmrabet	1000	RAM	
PL4	Henri alqui	3400	AF	
PL5	Ahmed Saadi	950	RAM	

Deux tables à mettre en jointure

Comp	nrue	rue	ville	nomComp
AF	122	Paris	Paris	Air France
RAM	7	BD des FAR	Casablanca	Royal Air Maroc
SING	1	Camparols	Singapour	Singapore AL

brevet	nom	nbVol	compa	chefPil
PL1	Pierre lamo	450	AF	PL4
PL2	Didier Linx	900	AF	PL4
PL3	Hamid Lmrabet	1000	RAM	
PL4	Henri alqui	3400	AF	
PL5	Ahmed Saadi	950	RAM	



Equijointure

- **Requête 1:** l’identité des pilotes de la compagnie de nom 'Air France' ayant plus de 500 heures de vol?

- **Requête 2:** les coordonnées des compagnies qui embauchent des pilotes de moins de 500 heures de vol

```
SELECT brevet, nom
FROM Pilote, Compagnie
WHERE comp=compa
AND nomComp = 'Air France‘ AND nbHVol >
500;

SELECT cpg.nomComp, cpg.nrue,
cpg.rue, cpg.ville
FROM Pilote pil, Compagnie cpg
WHERE cpg.comp=pil.compa
AND pil.nbHVol < 500;
```

```
SELECT brevet, nom
FROM Compagnie

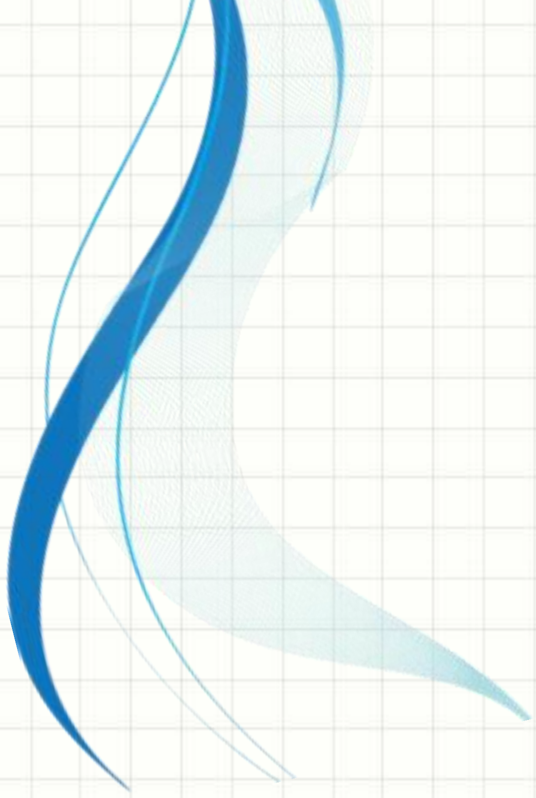
JOIN Pilote ON comp = compa

WHERE nomComp = 'Air France'
AND nbHVol > 500;
```

```
SELECT nomComp, nrue, rue, ville
FROM Compagnie

INNER JOIN Pilote ON comp = compa

WHERE nbHVol > 500;
```



Autojointure

Autojointure

brevet	nom	nbVol	compa	chefPil
PL1	Pierre lamo	450	AF	PL4
PL2	Didier Linx	900	AF	PL4
PL3	Hamid Lmrabet	1000	RAM	
PL4	Henri alqui	3400	AF	
PL5	Ahmed Saadi	950	RAM	

- **Requête 3:** l’identité des pilotes placés sous la responsabilité des pilotes de nom 'Alqui‘?
- **Requête 4:** la somme des heures de vol des pilotes placés sous la responsabilité des chefs pilotes de la compagnie de nom 'Air France‘?

```sql
SELECT p1.brevet, p1.nom
FROM Pilote p1, Pilote p2
WHERE p1.chefPil = p2.brevet
AND p2.nom LIKE '%Alqui%';
```

```sql
SELECT SUM(p1.nbHVol)
FROM Pilote p1, Pilote p2,
Compagnie cpg
WHERE p1.chefPil = p2.brevet
AND cpg.comp = p2.compa
AND cpg.nomComp = 'Air France';
```

```
SELECT p1.brevet, p1.nom
FROM Pilote p1

JOIN Pilote p2
ON p1.chefPil = p2.brevet

WHERE p2.nom LIKE '%Alqui%';
```

```
SELECT SUM(p1.nbHVol)
FROM Pilote p1

JOIN Pilote p2
ON p1.chefPil = p2.brevet
      JOIN Compagnie
      ON comp = p2.compa

WHERE nomComp = 'Air France';
```

Jointures procédurales (Sous-requêtes)

Notre exemple avec les sous requêtes

Compagnie

Comp	nrue	rue	ville	nomComp
AF	122	Paris	Paris	Air France
RAM	7	BD des FAR	Casablanca	Royal Air Maroc
SING	1	Camparols	Singapour	Singapore AL

Pilote

brevet	nom	nbVol	compa	chefPil
PL1	Pierre lamo	450	AF	PL4
PL2	Didier Linx	900	AF	PL4
PL3	Hamid Lmrabet	1000	RAM	
PL4	Henri alqui	3400	AF	
PL5	Ahmed Saadi	950	RAM	

- **Requête 1: :** l’identité des pilotes de la compagnie de nom 'Air France' ayant plus de 500 heures de vol?

```
SELECT brevet, nom FROM Pilote
WHERE

compa = (SELECT comp
         FROM Compagnie
         WHERE nomComp = 'Air France')


AND nbHVol>500;
```

Sous-interrogations multi-lignes : IN, ALL & ANY

Les opérateurs: IN, ALL,ANY

Exemple à traiter

immat	typeAv	nbHvol	compa
A1	A320	1000	AF
A2	A330	1500	AF
A3	A320	550	RAM
A4	A340	1800	RAM
A5	A340	200	AF
A6	A330	100	AF

VOIR AUSSI TD ?

Vous avez des questions ?

MERISE
Modélisation des systèmes d'information

Cycle de vie

« La qualité du processus de fabrication est garante de la qualité du produit »

- Pour obtenir un logiciel de qualité, il faut en maîtriser le processus d'élaboration
  - La vie d'un logiciel est composée de différentes étapes
  - La succession de ces étapes forme le cycle de vie du logiciel
  - Il faut contrôler la succession de ces différentes étapes

Etude de faisabilité

- Déterminer si le développement proposé vaut la peine d'être mis en œuvre, compte tenu de attentes et de la difficulté de développement
  - Etude de marché : Déterminer s'il existe un marché potentiel pour le produit.

Spécification

- Déterminer les fonctionnalités que doit posséder le logiciel
  - Collecte des exigences : obtenir de l'utilisateur ses exigences pour le logiciel
  - Analyse du domaine : déterminer les tâches et les structures qui se répètent dans le problème

Organisation du projet

- Déterminer comment on va développer le logiciel
  - Analyse des coûts : établir une estimation du prix du projet
  - Planification : établir un calendrier de développement
  - Assurance qualité du logiciel : déterminer les actions qui permettront de s'assurer de la qualité du produit fini
  - Répartition des tâches : hiérarchiser les tâches et sous-tâches nécessaires au développement du logiciel

Conception

- Déterminer la façon dont dont le logiciel fournit les différentes fonctionnalités recherchées
  - Conception générale
    - Conception architecturale : déterminer la structure du système
    - Conception des interfaces : déterminer la façon dont les différentes parties du système agissent entre elles
  - Conception détaillée : déterminer les algorithmes pour les différentes parties du système

Implémentation

- Ecrire le logiciel

Tests

- Essayer le logiciel sur des données d'exemple pour s'assurer qu'il fonctionne correctement
  - Tests unitaires : faire tester les parties du logiciel par leurs développeurs
  - Tests d'intégration : tester pendant l'intégration
  - Tests de validation : pour acceptation par l'acheteur
  - *Tests système :* tester dans un environnement proche de l'environnement de production
  - *Tests Alpha :* faire tester par le client sur le site de développement
  - *Tests Bêta :* faire tester par le client sur le site de production
  - *Tests de régression :* enregistrer les résultats des tests et les comparer à ceux des anciennes versions pour vérifier si la nouvelle n'en a pas dégradé d'autres

Livraison

- Fournir au client une solution logicielle qui fonctionne correctement
  - Installation : rendre le logiciel opérationnel sur le site du client
  - Formation : enseigner aux utilisateurs à se servir du logiciel
  - Assistance : répondre aux questions des utilisateurs

Maintenance

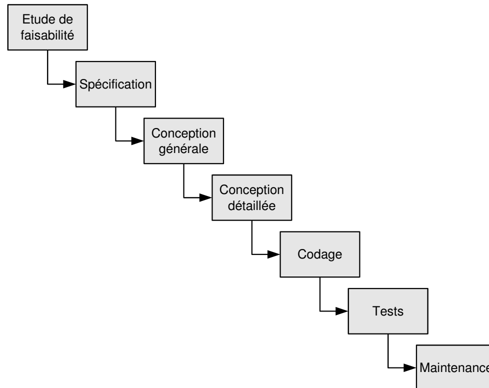
- Mettre à jour et améliorer le logiciel pour assurer sa pérénité
- Pour limiter le temps et les coûts de maintenance, il faut porter ses efforts sur les étapes antérieures

	Répartition effort dév.	Origine des erreurs	Coût de la maintenance
Définition des besoins	6%	56%	82%
Conception	5%	27%	13%
Codage	7%	7%	1%
Intégration Tests	15%	10%	4%
Maintenance	67%		

Modèles linéaires et incrémentaux

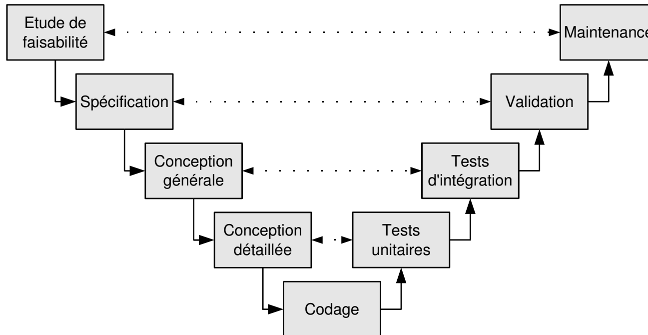
- Modèles linéaires
  - cascade
  - modèle en V
  - ...
- Modèles non linéaires
  - prototypage
  - modèles incrémentaux
  - modèle en spirale
  - ...

Le cycle de vie en « Cascade »



- Adapté pour des projets de petite taille, et dont le domaine est bien maîtrisé

Le cycle de vie en « V »



- Adapté pour des projets dont le domaine est bien maîtrisé

Le prototypage

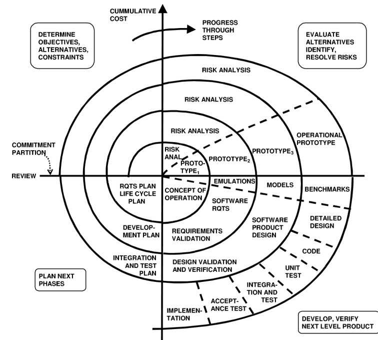
- Prototype : version d'essai du logiciel
  - Pour tester les différents concepts et exigences
  - Pour montrer aux clients les fonctions que l'on veut mettre en œuvre
- Lorsque le client a donnné son accord, le développement suit souvent un cycle de vie linéaire
- Avantages : Les efforts consacrés au développement d'un prototype sont le plus souvent compensés par ceux gagnés à ne pas développer de fonctions inutiles

Le modèle incrémental de Parnas

1. Concevoir et livrer au client un sous-ensemble minimal et fonctionnel du système
2. Procéder par ajouts d'incréments minimaux jusqu'à la fin du processus de développement
3. Avantages : Meilleure intégration du client dans la boucle, produit conforme à ses attentes

Le modèle en Sprirale de Boehm

- Un modèle mixte
- A chaque cycle, recommencer :
  1. Consultation du client
  2. Analyse des risques
  3. Conception
  4. Implémentation
  5. Tests
  6. Planification du prochain cycle



- Avantages : meilleure maîtrise des risques, mais nécessite une (très) grande expérience

Méthode : une démarche et un formalisme

- Démarche : succession d'étapes pour
  - Mieux maîtriser le déroulement d'un projet
  - Meilleure visibilité pour les utilisateurs sur certains résultats intermédiaires et garantir que le résultat final sera celui attendu
- Formalisme défini par:
  - Un langage formel
  - Un langage semi-formel généralement graphique
  - Un langage naturel
- Fonction :
  - Représenter le monde réel tel qu'il est perçu par le concepteur
  - Outil de communication entre informaticiens et utilisateurs
  - Constitué par un ensemble de modèles permettant d'assurer une bonne compréhension des besoins des utilisateurs

Modèles

- Représentation abstraite de la réalité qui exclut certains détails du monde réel
- Permet de réduire la complexité d'un phénomène en éliminant les détails qui n'influencent pas son comportement significatif
- Reflète ce que le concepteur croit important pour la compréhension et la prédiction du phénomène modélisé, les limites du phénomène modélisé dépendent des objectifs du modèle

MERISE

Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise

- Méthode Eprouvée pour Retarder Indéfiniment la Sortie des Etudes
- MEthode pour Rassembler les Idées Sans Effort
  - Sutout lorsqu'on utilise un AGL

Approche Données / Traitements

- Pour étudier et développer l'informatique d'une organisation, il est nécessaire de connaître:
  - comment elle réagit à une sollicitation externe
  - quelle est la structure des informations qu'elle utilise
- MERISE modélise cette connaissance de manière duale :
  - Modèles des Traitements (réaction aux événements...)
  - Modèles des Données (vocabulaire de la structure...)
  - Les 2 aspects sont complémentaires, synchronisés et validés entre eux

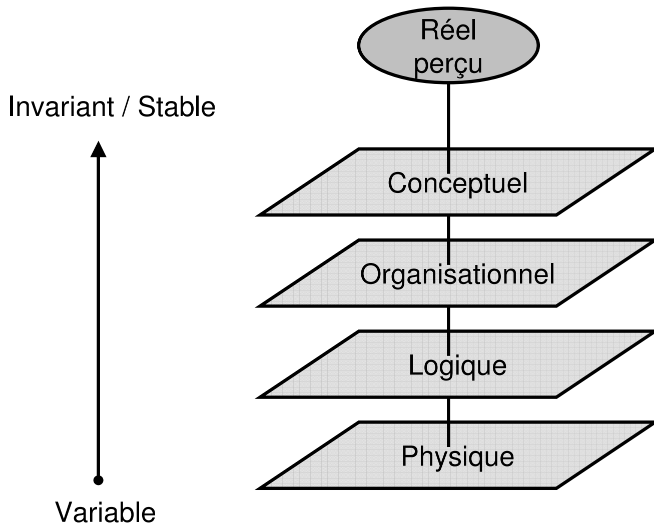
Niveaux d'abstraction

- Pour chacun des problèmes de modélisation (données / traitements)
  - Procéder de manière progressive...
  - ... du plus stable au plus technique

Niveaux d'abstraction

- Niveau Conceptuel
  - Ce qu'il faut faire
  - Quoi ?
- Niveau Organisationnel
  - La manière de faire
  - Pour les traitements
- Niveau Logique
  - Choix des moyens et ressources
  - Pour les données
- Niveau Physique
  - Les moyens de le faire
  - Comment ?

Niveaux d'abstraction



Exemples de niveaux d'abstraction

- Conceptuel
  - Le client effectue une demande de service à la compagnie pour assurer son véhicule. Cette dernière lui propose un devis
- Organisationnel
  - Un client effectue une demande de service à l'agence de son choix, par courrier, pour assurer un véhicule. Un agent de service concerné, si le client est fiable (consultation d'un fichier central inter assurances), prend contact par téléphone pour une visite à domicile (après 17 heures) afin d'examiner plus précisément ses besoins et établir un devis
- Physique
  - Le fichier central inter assurances est accessible par internet. Les agences sont connectées au siège de la compagnie par liaison ADSL. Chaque agence dispose de micro-ordinateurs de type PC et peut traiter ses données en local grâce au SGBD Access

Le niveau Conceptuel

- Exprime les choix fondamentaux de gestion, les objectifs de l'organisation
- Décrit les invariants de l'organisation
  - Le métier de l'organisation
- Définit
  - Des activités
  - Des choix de gestion
  - Des informations
- Indépendamment
  - Des aspects organisationnels
  - Des aspects techniques de mise en oeuvre
- Du point de vue
  - Des traitements: objectif, résultat, règle de gestion, enchaînement
  - Des données: signification, structure, liens

Le niveau Organisationnel

- Exprime les choix organisationnels de ressources humaines et matérielles
- Définit:
  - La répartition géographique et fonctionnelle des sites de travail (du point de vue des données et des traitements)
  - Le mode de fonctionnement : temps réel ou temps différé
  - La répartition du travail homme/machine (degré et type d'automatisation)
  - Les postes de travail et leur affectation,
  - La volumétrie des données
  - La sécurité des données
- Indépendamment des moyens de traitement et de stockage de données actuels ou futurs
- Les opérations conceptuelles vont être décomposées au niveau organisationnel en une ou plusieurs opérations organisationnelles

Le niveau Logique

- Exprime la forme que doit prendre l'outil informatique pour être adapté à l'utilisateur, à son poste de travail
- Indépendamment de l'informatique spécifique, des langages de programmation ou de gestion des données
- Introduit la notion d'outils en tant que fonction réutilisable
- Décrit
  - Le schéma de la base de données (relationnel, hiérarchique ou réseau), cad les caractéristiques du mode de gestion des données
  - La répartition des D sur les différentes unités de stockage
  - Les volumes par unité de stockage
  - L'optimisation des coûts induits par le mode de gestion

Le niveau Physique

- Traduit les choix techniques et la prise en compte de leurs spécificités
- Répond aux besoins des utilisateurs sur les aspects logiciels et matériels.
- Définit complètement:
  - Les fichiers, les programmes
  - L'implantation physique des données et des traitements
  - Les ressources à utiliser
  - Les modalités de fonctionnement

Les modèles au niveau Conceptuel

- Le Modèle Conceptuel des Données (MCD)
  - Description des données et des relations en termes de
    - Entité ou Individu
    - Relation ou Association
    - Propriétés ou d'Attributs
- Le Modèle Conceptuel des Traitements (MCT)
  - Description de la partie dynamique du SI en termes de
    - Processus
    - Opérations

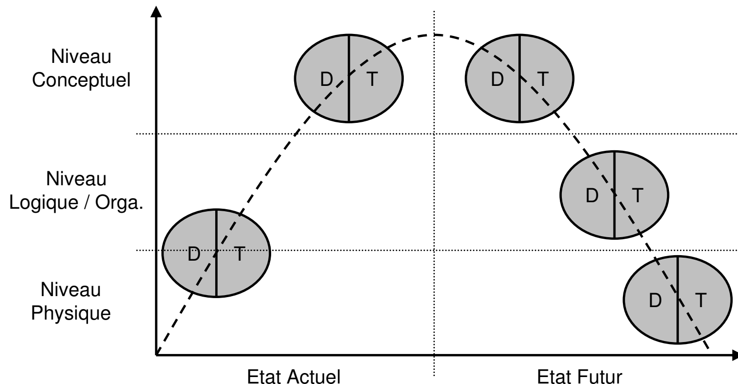
Les modèles aux niveaux Organisationnel et Logique

- Le Modèle logique de donnée (MLD)
  - Le modèle « CODASYL » si une orientation base de données réseau est choisie
  - Le modèle « relationnel » si une orientation base de données relationnelle est choisie
  - Le modèle « hiérarchique »
- Le Modèle Organisationnel des Traitement (MOT)
  - Permet de représenter par procédure les phases et les tâches effectuées par chaque poste de travail

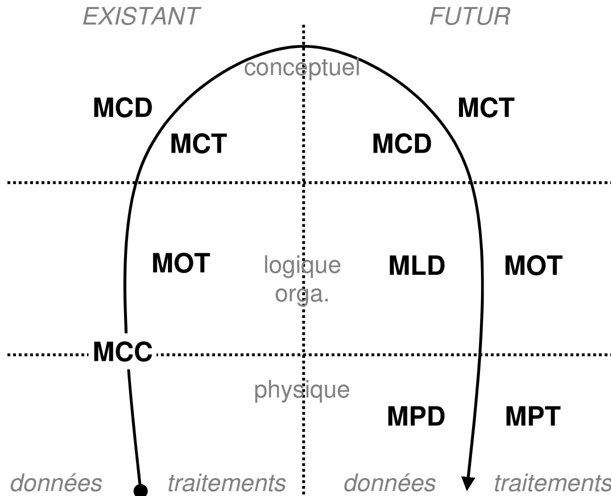
Les Modèles au niveau Physique

- Le Modèle Physique des Données (MPD)
  - Spécifie les organisations physiques de données
- Le Modèle Physique des Traitements (MPT)
  - Décrit les traitements réalisés pour chaque transaction (temps réel) ou chaque unité de traitement (temps différé)

Processus de développement



Modèles successifs produits



Organisation du projet

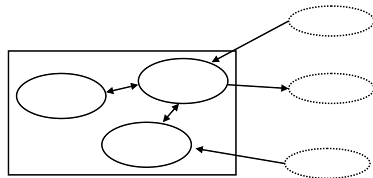
- Par groupe de 5 étudiants : analyse complète du cas proposé
- Pour chaque scéance de TD
  - Conception du modèle demandé pour la scéance en question
  - A la fin de chaque scéance, l'enseignant collecte votre travail
  - Au début de chaqué scéance
    - L'enseignant vous rend le travail de la scéance précédente corrigé
    - Vous prenez en compte les corrections pour les étapes ultérieures
- La note finale est la somme des notes partielles obtenues à chaque scéance

Echéancier

- Semaine :
  1. Compte rendu d'entretiens et MCC
  2. MCT
  3. VED pour chaque opération
  4. MCD en 3ème forme normale
  5. MOT
  6. MPD
  7. Génération d'une base de données
  8. Synthèse

Modèle Conceptuel de Communication (MCC)

- Représente, au niveau conceptuel, les échanges d'information entre les acteurs



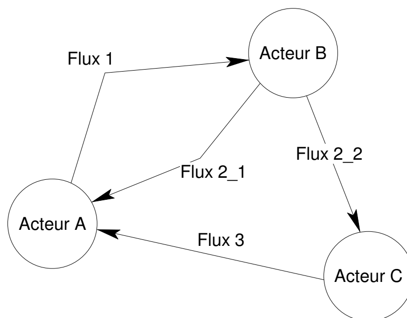
- Première étape d'une étude de l'existant, pour modéliser les habitudes de travail dans l'organisation concernée
  - Délimiter le domaine étudié
  - Réduire la complexité en identifiant des sous problèmes traités individuellement
  - Identifier les acteurs externes et internes
  - Modéliser les échanges d'informations entre les différents acteurs

Acteurs

- Représenté par un cercle libellé par le nom de l'acteur
- L'acteur représente une unité active intervenant dans le fonctionnement d'un système opérant. Il peut
  - Etre stimulé par des flux d'information
  - Transformer et émettre des flux d'information
- Un acteur « fait quelque chose », il est actif
  - Ex : Service comptabilité, Guichet ...
- Un acteur est un rôle plutôt qu'une personne physique (« Direction » et pas « Jean-Claude »)
  - Il peut être pertinent de modéliser séparément deux fonctions assumées par une même personne physique
- On distingue les acteurs internes et externes

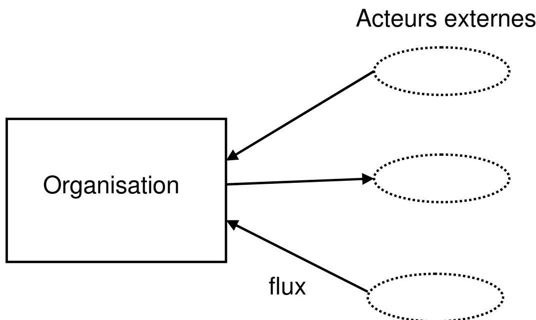
Flux d'information

- Représenté par une flèche entre deux acteurs, étiquetée par le nom du flux
- Echange d'informations entre deux acteurs
  - Ex : documents, appels téléphoniques, données informatiques



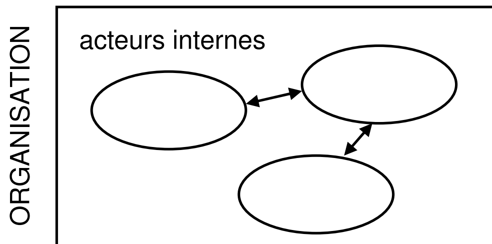
Acteurs externes

- Eléments externes avec lesquels le système échange des flux d'information
  - Ex : clients, fournisseurs...



Acteurs internes

- Acteurs faisant partie du système d'information étudié
  - Ex : guichet, service informatique...
- Si le système est complexe, on peut considérer un acteur interne comme un sous-domaine et détailler ce sous-domaine dans un nouveau MCC



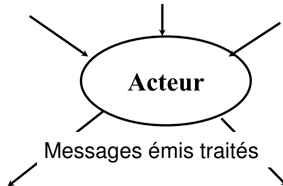
Modèle Conceptuel des Traitements (MCT)

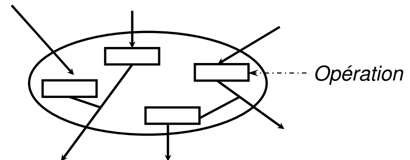
- Représente formellement les activités exercées par le domaine (à la base de la connaissance du SI)
- Repose sur la prise en compte des échanges (flux) du domaine avec son environnement
- S'effectue en faisant abstraction de l'organisation et des choix technologiques

La définition des interactions du domaine avec son environnement prime sur la manière dont on assurera ces activités

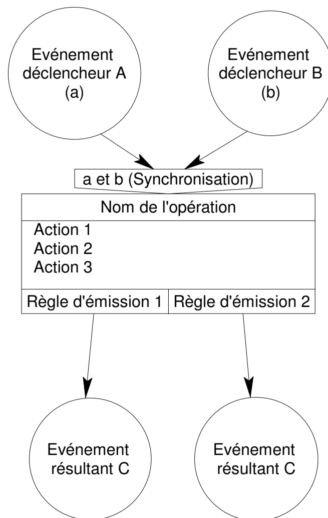
MCC et MCT

- Le MCT est un « zoom » sur le MCC
  - Dans les MCC, on représente les messages échangés entre acteurs
  - Dans les MCT, on représente comment un acteur de l'organisation réagit quand il reçoit ce message et quelle opération il effectue

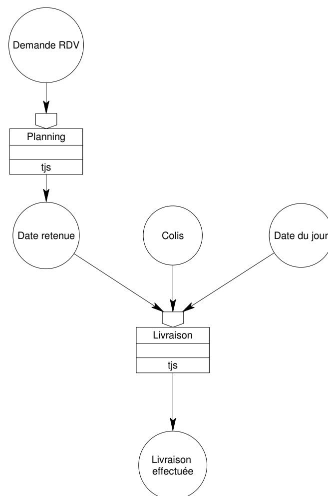




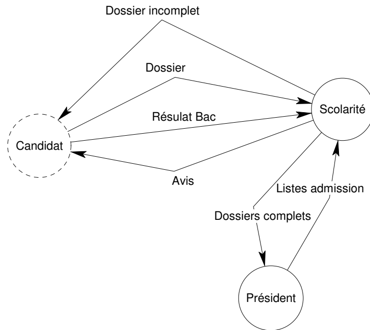
Modèle de MCT

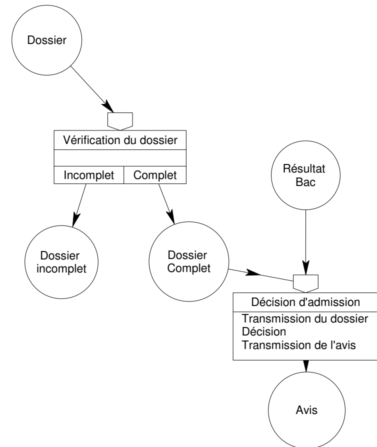


Exemple de MCT



Passage du MCC au MCT

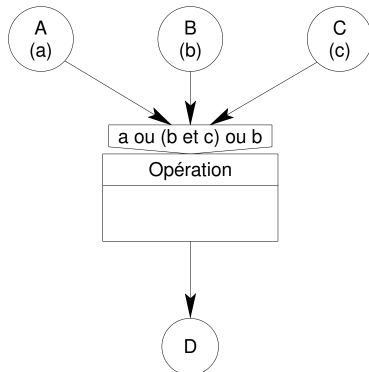




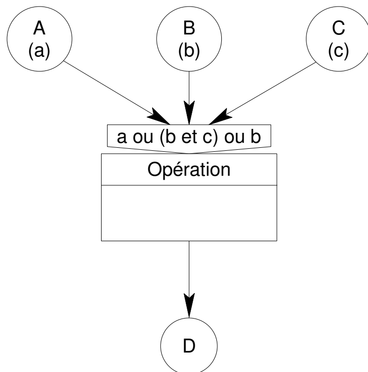
Erreurs de modélisation fréquentes

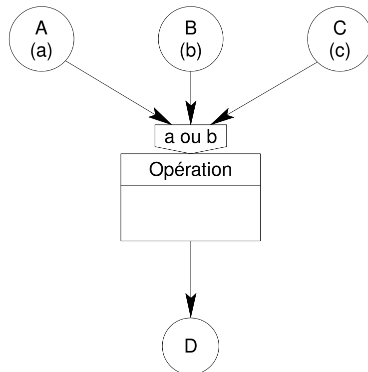
- Règles d'émission : elles doivent
  - Etre mutellement exclusives : deux règles de la même opération ne peuvent pas être vraies en même temps
  - Couvrir tous les cas possibles
- Ne pas répéter les actions et les événements résultants
- Problèmes de synchronisation
  - Il faut simplifier les synchonisations
- Problèmes structurel
  - Il faut éviter les chaînes d'opérations et les événements internes

Simplification des synchronisations



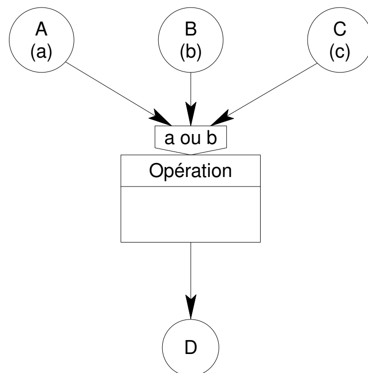
Simplification des synchronisations





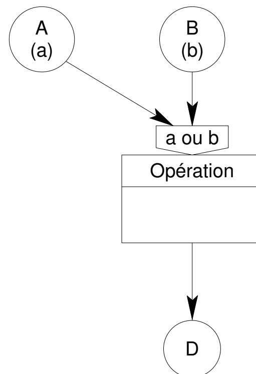
Simplification des synchronisations

La simplification a mis en évidence que C n'était pas nécessaire

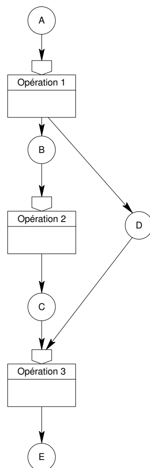


Simplification des synchronisations

La simplification a mis en évidence que C n'était pas nécessaire

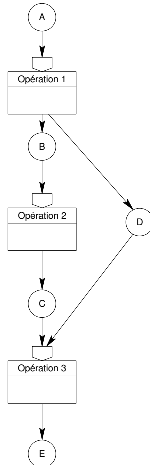


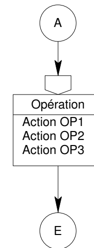
Réduction des chaînes d'opérations



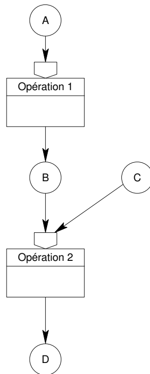
- De A à E, les opérations s'enchaînent de manière systématique
- On supprime les événements internes B, C et D

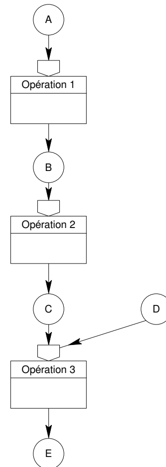
Réduction des chaînes d'opérations





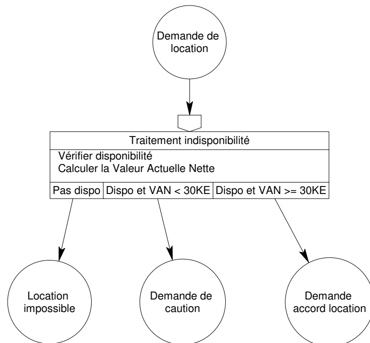
Autres exemples





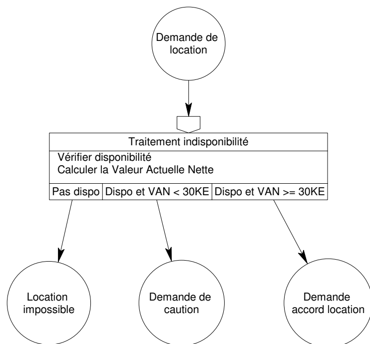
Chaînes à réduire à une seule opération

Cas d'introduction d'événements internes

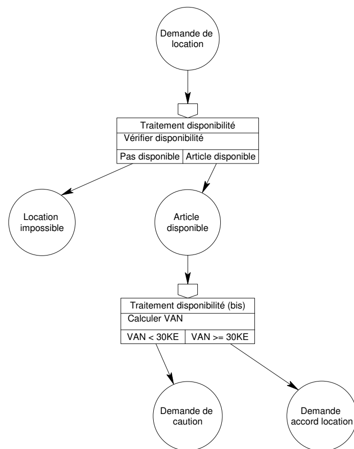


Calculer la VAN ne se fait pas en cas d'indisponibilité

Cas d'introduction d'événements internes



Calculer la VAN ne se fait pas en cas d'indisponibilité



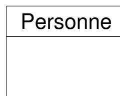
Modèle Conceptuel des Données

- Modèle Entité / Association
  - Souvent nommé Entité-Relation
- Repose sur les concepts de
  - Entités
  - Associations
  - Propriétés
- Permet de décrire un ensemble de données relatives à un domaine défini afin de les intégrer ensuite dans une Base de Données

Entité et entité type

- Entité : Une entité est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement
  - Ex : Jean-Claude, Momo, Ma Voiture, Son 4x4, l'Île de France, la Bretagne
- Entité type : Une entité type est la représentation commune que l'on adopte pour des entités qui possèdent les mêmes caractéristiques
  - Ex : Personne, Voiture, Région

Une entité est une *occurence* d'une entité type (ou instance)



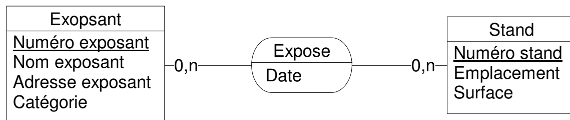
Propriété (ou attribut)

- Propriété : caractéristique associée à une entité type
  - Ex : L'âge d'une personne, la puissance d'une voiture, le numéro d'un produit...
  - On associe un *domaine* à chaque propriété, qui définit l'ensemble des valeurs possibles que peut prendre la propriété
- Valeur : Valeur que prend une propriété (à l'intérieur du domaine) pour une entité particulière
  - Ex : 28 ans pour l'âge de Jean-Claude, 150cv pour la puissance de son 4x4

Personne
Nom Prénom

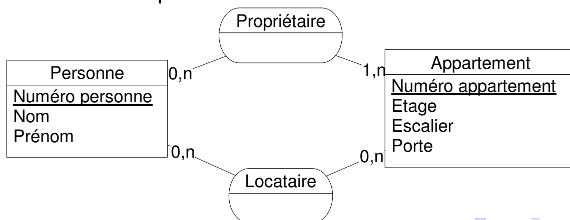
Association et association type

- Association : lien entre plusieurs entités
  - Ex : Le mariage de Momo et de Jeanne, celui de Jean-Claude et d'Eglantine
- Association type : représentation d'un ensemble de relations qui possèdent les mêmes caractéristiques, lien entre plusieurs entités type
  - Ex : Le mariage de deux personnes
- Une association type peut avoir des propriétés



Association et association type

- Association : lien entre plusieurs entités
  - Ex : Le mariage de Momo et de Jeanne, celui de Jean-Claude et d'Eglantine
- Association type : représentation d'un ensemble de relations qui possèdent les mêmes caractéristiques, lien entre plusieurs entités type
  - Ex : Le mariage de deux personnes
- Il peut y avoir plusieurs associations type liant les mêmes entités si la sémantique est différente



Abus de langage

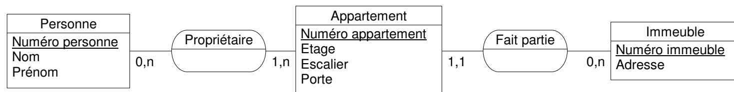
- Souvent, on parle d'« Entité » à la place d'« Entité Type ». Dans la suite, comme c'est d'usage, nous utiliserons les termes :
  - *Entité* pour *entité type*
  - *Occurence d'entité* pour *entité*
- De même, on utilise souvent « Association » plutôt que « Association Type ». Dans la suite, comme c'est d'usage, nous utiliserons les termes :
  - *Association* pour *Association type*
  - *Occurence d'association* pour *Association*

Identifiants

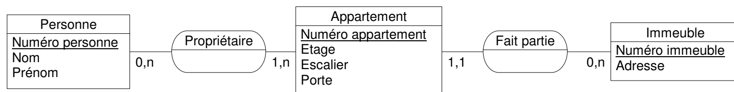
- Identifiant : une ou plusieurs propriétés d'une entité ou d'une association qui ont une valeur unique pour chaque occurence de l'entité ou de l'association
  - Ex : Le numéro de SECU d'une personne, le numéro d'immatriculation d'une voiture...
  - On souligne les identifiants d'une entité
  - L'identifiant d'une association est un sous-ensemble des identifiants des entités liés

Cardinalités

- Cardinalité d'une association : le nombre de fois minimal et maximal qu'une occurence d'une des entités associée peut intervenir dans l'association
  - Ex : un client peut commander entre 1 et n produits



Cardinalités



- Cardinalité minimale
  - 0 si une occurence de l'entité peut exister tout en n'intervenant dans aucune occurence de l'association
  - 1 si une occurence de l'entité ne peut exister que si elle intervient dans au moins une occurence de l'association
  - n : cas rare à éviter
- Cardinalité maximale
  - 1 si une occurence de l'entité ne peut pas être impliquée dans plus d'une occurence de l'association
  - n si une occurence de l'entité ne peut être impliquée dans plus d'une occurence de l'association

Dimension d'une association

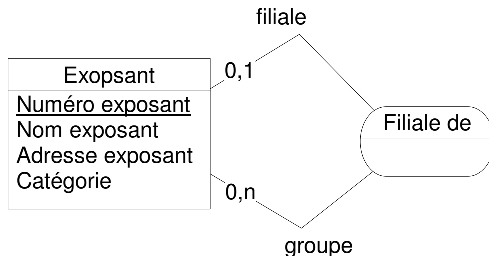
- Dimension : Nombre de « pattes » de l'association
  - Binaire, *ternaire* ou n-aire

Types d'asociations

- En fonction des cardinalités
  - 1:1 si toutes la cardinalités maximales valent 1
  - 1:n s'il existe au moins une cardinalité maximale à n et une à 1
  - n:m si toutes la cardinalités maximales valent n

Associations réflexives

- Association réflexive : Une association dont plusieurs « pattes » lient la même entité. Dans ce cas, plusieurs occurences de la même entité seront associées



- On peut libeller chaque « pattes » par son rôle dans l'association

Remarques

- Il est parfois difficile de faire un choix entre entité et association
  - Ex : Un mariage est-il une association entre deux personnes ou une entité pour lequel on veut conserver un numéro, une date, un lieu, etc. et que l'on souhaite manipuler en tant que tel ?
  - Souvent, le contexte aide à décider
    - Lorsqu'on ne parvient pas à trouver d'identifiant pour une entité, il faut se demander s'il ne s'agit pas en fait d'une association. Si ce n'est pas le cas, un identifiant arbitraire numérique entier peut faire l'affaire
    - Lorsque toutes les pattes d'une association portent la cardinalité 11, il faut se demander si ce type-association et les types-entités liés ne décrivent pas en fait un seul type-entité

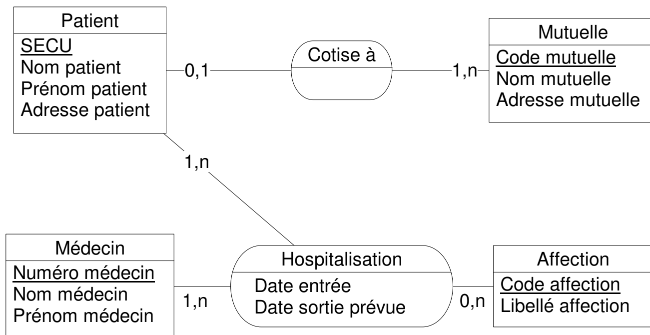
Cohérence entre données et traitements

- A chaque opération, on associe un MCD partiel : une Vue Externe des Données
  - On s'assure ainsi que toutes les données nécessaires sont représentées
- Le MCD global est l'union de toutes les VED
- Pour chaque élément du MCD global, on vérifie que celui-ci est utilisé dans au moins une opération
  - On s'assure ainsi que seules les données nécessaires sont représentées
- On s'appuie souvent sur des documents existants pour réaliser les VED

Dépendances fonctionnelles

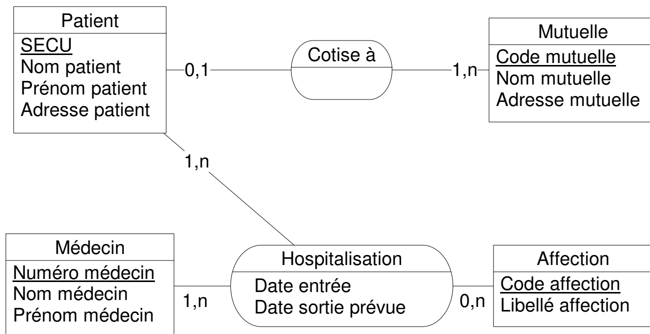
- Une propriété (ou un groupe de propriétés) Y dépend fonctionellement d'une autre propriété (ou groupe de propriétés) X si
  - Etant donné une valeur de X, il lui correspond une valeur unique de Y. On note
  - $X \rightarrow Y$ (X détermine Y)
- Cette relation est transitive : si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$
  - Cependant, on ne représente que les DF élémentaires

Dépendances fonctionnelles



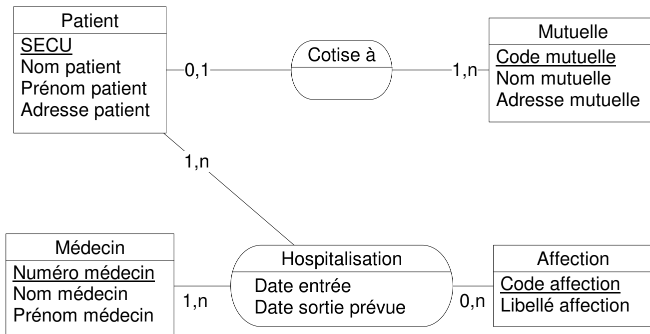
- Les propriétés non identifiantes d'une entité dépendent fonctionnellement de l'ensemble des identifiants
    - Ex : SECU → NomPatient, PrénomPatient, AdressePatient

Dépendances fonctionnelles



- L'identifiant d'une association de type nm dépend fonctionnellement des identifiants des entités liées
  - Ex : SECU, NuméroMédecin, CodeAffection → DateEntrée, DateSortie

Dépendances fonctionnelles



- Une cardinalité 11 ou 01 est la source d'une dépendance fonctionnelle de l'identifiant du côté 11 vers l'autre côté de l'association
  - Ex : SECU → CodeMutuelle

1ère Forme Normale (1FN)

- Toutes les entités et les association possèdent un identifiant
- Aucune propriété n'est à valeurs multiples (popriétés atomiques)

1ère Forme Normale (1FN)

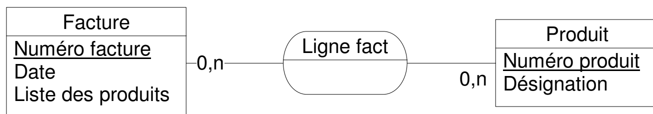
- Toutes les entités et les association possèdent un identifiant
- Aucune propriété n'est à valeurs multiples (popriétés atomiques)

Facture
Numéro
Date
Liste des produits

- Ici, « liste des produits » n'est pas atomique, c'est une liste

1ère Forme Normale (1FN)

- Toutes les entités et les association possèdent un identifiant
- Aucune propriété n'est à valeurs multiples (popriétés atomiques)



2e Forme Normale (2FN)

- Le modèle est en 1FN
- Toutes les DF entre les propriétés sont élémentaires
  - Toute propriété n'appartenant pas à une clé ne dépend pas seulement d'une partie de son identifiant identifiant
  - Les propriétés d'une entité ne doivent dépendre que de l'identifiant de l'entité et non d'une partie de cet identifiant

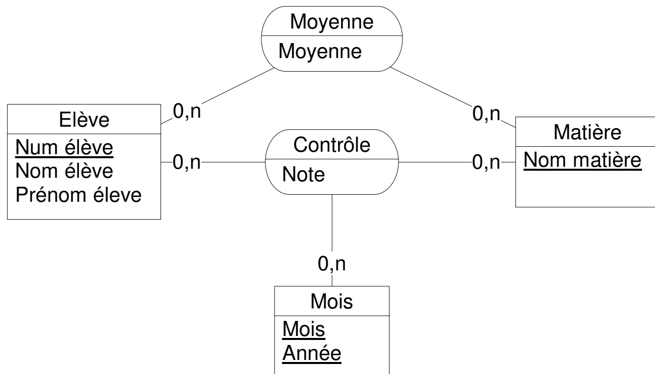
2e Forme Normale (2FN)

- Le modèle est en 1FN
- Toutes les DF entre les propriétés sont élémentaires

- Ici, d'après le schéma,
  NumEleve, NomMatière, Mois, Année $\rightarrow$ *Moyenne*
- Or, NumEleve et NomMatière suffisent

2e Forme Normale (2FN)

- Le modèle est en 1FN
- Toutes les DF entre les propriétés sont élémentaires



3e Forme Normale (3FN)

- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes
  - Les propriétés d'une entité doivent dépendre de l'identifiant de l'entité de manière directe
  - Toute propriété n'appartenant pas à un identifiant ne dépend pas d'un attribut non identifiant

3e Forme Normale (3FN)

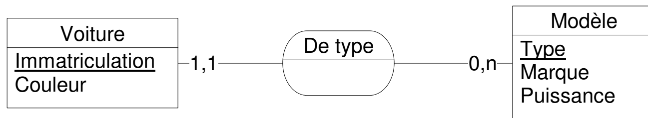
- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes

Voiture
<u>Immatriculation</u> Couleur Type Puissance Marque

- Or, *Type* $\longrightarrow$ *Marque, Puissance* alors que Type n'est pas un identifiant

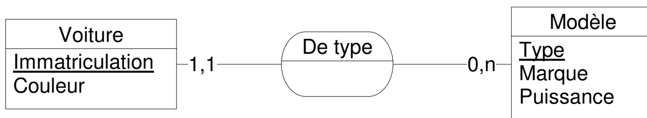
3e Forme Normale (3FN)

- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes



3e Forme Normale (3FN)

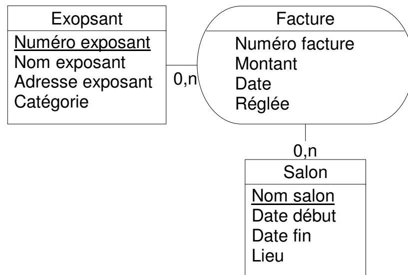
- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes



- Très bien mais si on voulait rajouter un numéro de facture...

3e Forme Normale (3FN)

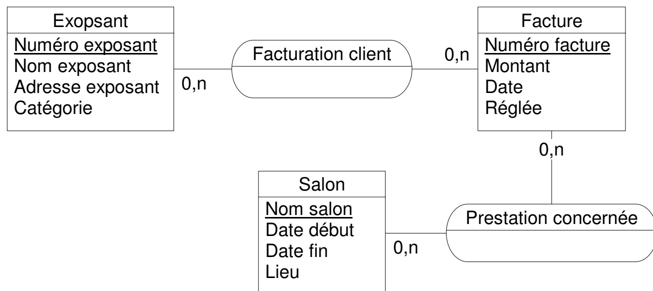
- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes



- Or, *NumFact* $\rightarrow$ *Montant, Date, Réglée* alors que NumFact n'est pas un identifiant

3e Forme Normale (3FN)

- Le modèle est en 2FN
- Toutes les DF entre les propriétés sont directes



Forme normale de Boyce-Codd (BCNF)

- Le modèle est en 3FN
- Les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles un identifiant détermine une propriété
  - Pour les identifiants composés de plusieurs propriétés, ces dernières ne doivent pas être dépendantes d'une autre propriété de l'entité (pour éviter les cycles de DF)

Forme normale de Boyce-Codd (BCNF)

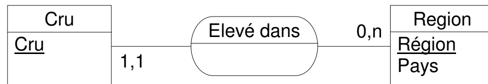
- Le modèle est en 3FN
- Les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles un identifiant détermine une propriété

Vin
Cru Pays Région

- Or, $Région \longrightarrow Pays$

Forme normale de Boyce-Codd (BCNF)

- Le modèle est en 3FN
- Les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles un identifiant détermine une propriété



- On a alors, *Cru* → *Region* et *Region* → *Pays*
- Attention : Même si elle peut être retrouvée par jointure, on a perdu la dépendance *Cru, Pays* → *Région*

Un MCD ne doit pas nécéssairement être en BCNF, il faut peser le pour et le contre avant de perdre des dépendances fonctionnelles

CONCEPTION DE LA BASE DE DONNÉES

Dépendances fonctionnelles

Formes normales

LA NORMALISATION & DÉPENDANCES FONCTIONNELLES

- Le processus de normalisation permet de construire des bases de données relationnelles en évitant les redondances et en préservant l’intégrité des données. Il est préférable de normaliser les relations au moins jusqu’à la troisième forme normale.
- La normalisation est basée sur les dépendances fonctionnelles (DF). E.F. Codd fut le premier à publier des écrits sur les DF

DÉPENDANCES FONCTIONNELLES: DÉFINITION

- Un attribut *b* dépend fonctionnellement d'un attribut *a* si à une valeur de *a* correspond au plus une valeur de *b*. La *dépendance fonctionnelle* est notée $a \longrightarrow b$.
- Le membre droit de l'écriture s'appelle le *dépendant,* le membre gauche s'appelle le *déterminant*.

DÉPENDANCES FONCTIONNELLES

- Exemple :
- l’écriture numPilote,jour $\longrightarrow$ nbHeuresVol est une DF, car à un couple (numPilote, jour) correspond au plus un nombre d’heures de vol;
- l’écriture numPilote $\longrightarrow$ nomPilote, fonction est équivalente aux écritures numPilote $\longrightarrow$ nomPilote et numPilote $\longrightarrow$ fonction qui sont deux DF. En conséquence
- numPilote $\longrightarrow$ nomPilote, fonction est une DF ;

DÉPENDANCES FONCTIONNELLES : DÉFINITION

- ***DF élémentaire***
  - Une DF *a,b→c* est *élémentaire* si ni *a→c*, ni *b→c* ne sont des DF.
- ***DF directe***
  - Une DF *a* → *c* est *directe* si elle n'est pas déduite par transitivité, c'est-à-dire s'il n'existe pas de DF *a* → *b* et *b* → *c*.

DÉPENDANCES FONCTIONNELLES

- Exemple

Considérons l’exemple qui fait intervenir les attributs suivants : (immat : numéro d’immatriculation d’un avion ; typeAvion : type de l’aéronef ; nomConst : nom du constructeur).

● La dépendance immat → nomConst n’est pas une DF directe.

● La dépendance immat → typeAvion est une DF directe.

● La dépendance typeAvion → nomConst est une DF directe.

DÉPENDANCES FONCTIONNELLES PROPRIÉTÉS :
« axiomes d'Armstrong » ou « règles d'inférence »

- *Réflexivité* : si *b* est un sous-ensemble de *a* alors $a \rightarrow b$ est une DF. Une autre écriture de la réflexivité (appelée « autodétermination ») $a \rightarrow a$ est une DF,
- *Augmentation* : si $a \rightarrow b$ est une DF, alors $a,c \rightarrow b,c$ est une DF.
- *Transitivité* : si $a \rightarrow b$ et $b \rightarrow c$ sont des DF, alors $a \rightarrow c$ est une DF.
- *Union :* si $a \rightarrow b$ et $a \rightarrow c$ sont des DF, alors $a \rightarrow b,c$ est une DF.
- *Pseudo-transitivité :* si $a \rightarrow b$ et $b,c \rightarrow d$ sont des DF, alors $a,b \rightarrow d$ est une DF.
- *Décomposition* : si $a \rightarrow b,c$ est une DF, alors $a \rightarrow b$ et $a \rightarrow c$ sont des DF.

Formes normales

- La normalisation permet de construire des schémas optimaux en terme d’intégrité et de cohérence des données stockées.

Formes normales :
Première forme normale

- Une relation est en *première forme normale* si chaque *n*-uplet contient une seule valeur par attribut.
- En d'autres termes, au niveau de l'extension d'une relation, à l'intersection d'une ligne et d'une colonne, on ne doit trouver qu'une et une seule valeur (qui peut être diverse : nombre, chaîne de caractères, date, image, etc.)

Formes normales : *Deuxième forme normale*

Une relation est en *deuxième forme normale* si

1. elle est en première forme normale,
2. et tout attribut n'appartenant pas à la clé primaire est en dépendance fonctionnelle élémentaire avec la clé.

Formes normales : *Troisième forme normale*

- Une relation est en *troisième forme normale* si elle respecte la deuxième forme normale et si les dépendances fonctionnelles entre la clé primaire et les autres attributs sont directes.